Developing a Reference Framework for Cybercraft Trust Evaluation

THESIS

Shannon E. M. Hunt, Captain, USAF

AFIT/GCS/ENG/08-11

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/08-11

Developing a Reference Framework for Cybercraft Trust Evaluation

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science (Computer Science)

Shannon E. M. Hunt, B.S.C.S

Captain, USAF

March 2008

AFIT/GCS/ENG/08-11

Developing a Reference Framework for Cybercraft Trust Evaluation

Shannon E. M. Hunt, B.S.C.S
Captain, USAF

Approved:

/signed/	06 Mar 2008
Lt Col J. Todd McDonald, Ph.D. (Chairman)	date
/signed/	06 Mar 2008
Maj Paul Williams, Ph.D. (Member)	date
/signed/	06 Mar 2008
Dr. Gilbert Peterson (Member)	date

AFIT/GCS/ENG/08-11

Abstract

It should be no surprise that Department of Defense (DoD) and U.S. Air Force (USAF) networks are the target of constant attack. As a result, network defense remains a high priority for cyber warriors. On the technical side, trust issues for a comprehensive end-to-end network defense solution are abundant and involve multiple layers of complexity. The Air Force Research Labs (AFRL) is currently investigating the feasibility of a holistic approach to network defense, called Cybercraft. We envision Cybercraft to be trusted computer entities that cooperate with other Cybercraft to provide autonomous and responsive network defense services. A top research goal related to Cybercraft centers around how we may examine and ultimately prove features related to this root of trust.

In this work, we investigate use-case scenarios for Cybercraft operation with a view towards analyzing and expressing trust requirements inherent in the environment. Based on a limited subset of functional requirements for Cybercraft in terms of their role, we consider how current trust models may be used to answer various questions of trust between components. We characterize generic model components that assist in answering questions regarding Cybercraft trust and pose relevant comparison criteria as evaluation points for various (existing) trust models. The contribution of this research is a framework for comparing trust models that are applicable to similar network-based architectures. Ultimately, we provide a reference evaluation framework for how (current and future) trust models may be developed or integrated into the Cybercraft architecture.

Acknowledgements

This work could not have been accomplished without the help of many people. I owe any creativity of thought to my Lord and Savior, Jesus Christ. Next, I want to thank my advisor, Lt. Col McDonald, for all his expertise and hard work in pushing me to accomplish this thesis despite my shortcomings. As well, thanks to my committee members Dr. Peterson and Maj Williams for their helpful insights during the whole process. I want to thank Mr. Giannelli of INOSC East Det 3 for his tremendous help in hashing out the requirements.

Thank you to my sister for help with editing at the 11^{th} hour and constant encouragement. Some days I really needed those breaks! I can't finish without a special thanks to my Mom and Dad, always having faith in me and trusting that I will succeed no matter what I do. And for all the support and encouragement they give me every day. I love you Mom and Dad!

Shannon E. M. Hunt

Table of Contents

Page

List of Figures

List of Tables

Table Page

Table Page

Developing a Reference Framework for Cybercraft Trust Evaluation

I. Introduction

The world is in the middle of the information age with almost anything easily accessible through the Internet. Attackers are everywhere exploiting this wealth of information and targeting U.S. military information systems. Working towards combating these new threats, the USAF redefined its mission in 2005 to include cyberspace - "deliver sovereign options for the defense of the United States of America and its global interests - to fly and fight in Air, Space, and Cyberspace." [9] The strategic vision for Cyberspace as a warfighting domain was furthered by Secretary of the Air Force Michael W. Wynne when he announced the creation of the Cyberspace Command (AFCYBER) [13]. Focusing on science and technology issues related to this domain, AFRL launched a research initiative geared to prepare for defense in this critical realm of cyberspace, termed Cybercraft. Just as aircraft platforms operate in air and carry a wide variety of payloads (bombs, missiles, electronic warfare pods, precision guided munitions), the term Cybercraft reflects the idea of generic platforms operating in cyberspace and executing a wide variety of payloads (patch verification, router configuration information, INFOCON policy enforcement, etc.).

The likely Cybercraft architecture consists of a machine-installed platform that executes mission-specific payloads: the platform represents a trusted component that provides command, control, and communication of cyber capabilities on host nodes while the payloads inherit trust from the platform and carry out various defensive missions and goals. Network defenders will use this architecture to accomplish specific tasks via single or multiple cooperating Cybercraft payloads. The architecture will incorporate hybrid trusted hardware/software/firmware components in various levels of interaction to support desired functionalities such as intrusion detection, anti-virus monitoring, network defense, and so forth. Cybercraft may eventually be deployed on up to one million nodes in order to provide commanders with a root of trust related to their high level strategic and operational network defense needs.

1.1 Research Motivation

1.1.1 Goals. The ability to model, measure, and verify the degree of trust a commander may place in Cybercraft remains a crucial research question that must be adequately characterized before this future architecture becomes a reality. Although trust itself has a multitude of meanings, in this work we consider how to synthesize and express the nature of trust in the future Cybercraft environment based on expected operational requirements. We further characterize generic model components that will help answer questions regarding Cybercraft trust and pose relevant comparison criteria as evaluation points for various (existing) trust models. We also introduce a novel approach to synthesize trust relationships iteratively based on use case analysis, attack tree threat modeling and operational/mission level task breakdown.

1.2 Research Contribution

1.2.1 Requirements. We provide a unique and revolutionary approach to requirements definition based on: use case analysis, attack/defense trees and mission level task breakdown. Attack trees are a way to visualize attacks on our networks as well as possible defense approaches to these attacks. A use case is typically described as a text-based step-by-step breakdown of the interaction between a user and a system. Mission level task breakdown is the process of splitting a high level goal into smaller, more manageable pieces. Our initial set of requirements helps further the goal of creating a Cybercraft prototype.

1.2.2 Reference Framework for Trust Evaluation. Our contribution to the area of trust is creating a way to characterize generic model components to establish trust boundaries within the Cybercraft domain. Given specific requirements specification, we derive attributes and desired properties of trust models which articulate, express, and evaluate commanders trust. We provide specific correlation between abstract trust models and the Cybercraft trust problem related to specific system requirements. Furthermore, we implement and analyze specific models to demonstrate the utility of trust expression within the context of Cybercraft. Given three specific trust models (hTrust, VTrust, P2P), we illustrate and analyze the nature of transitive trust decisions reflected by components of the Cybercraft architecture. Finally, we define a reference framework for evaluating existing and future trust models as well as provide specific measures for analyzing transitive trust relationships in view of the Cybercraft platform and its root of trust.

1.3 Thesis Organization

This document is divided into seven chapters. Chapter II presents Cybercraft and trust issues in greater detail. Chapter III presents our approach to requirements gathering using attack/defense trees, use cases and mission level task breakdown. Chapter IV characterizes the trust models and further explains how to evaluate them. Chapter V sets up the scenarios. Chapter VI walks through and analyzes the results and Chapter VII summarizes our contributions and give recommendations for future work.

Thesis Statement: This research examines the trust relationships throughout the Cybercraft architecture and develops a clear way of gathering requirements by using attack trees, use cases and mission level task breakdown.

Results: This research creates an initial set of requirements for the Cybercraft Domain (Chapter III, Section 3.3). These requirements, along with three trust models (hTrust, VTrust, P2P), create a reference framework for Cybercraft trust evaluation. Analysis of the three trust models concludes no model rises above another and, as is, none are suitable for potential use within the Cybercraft architecture.

This research first creates an initial set of requirements for the Cybercraft domain (Chapter III, Section 3.3). Next, we create a referece framework for trust model evaluation with possible application to Cybercraft. Three models in particular (hTrust, VTrust, P2P) were evaluated to create the reference framework (Chapter VI, Section 6.3). Our analysis of the three models concludes that none, in their current state, are applicable for Cybercraft. Using the reference framework as a guide for trust model requirements, a combination of the best attributes of the three models is a possibility.

II. Related Work

Defining trust is elusive as no one definition rises above another. One of the reasons is that trust is more of a social issue as opposed to the technical view many in the research community have.

Although trust is used everywhere, Gollmann argues that just using the word (trust) in a system or project is dangerous because of its manifold and sometimes contradictory meanings [10]. It is an overloaded term that hinders the clarity and precision that is sought after in technical fields. Nonetheless, it expresses a quality that military commanders make quite frequently: an objective dependability (whether by mathematical proof or demonstrated testing) that a system will perform according to its specifications, even though negative consequences can occur. Next, we discuss the specific ideas of trust that apply to Cybercraft in context to the envisioned architecture.

2.1 Trust

Many authors have attempted to define trust. Gambetta [8] laid the foundation for the definition of trust as a social concept that is subjective and context-dependent. Cahill [4] elaborated further to add attributes such as self-preserving and self-amplifying, among others. In more general terms, trust is defined as the measure of trustworthiness that relies on whatever evidence is provided or implied [3]. Trust plays a key role in system development and we consider it an essential concept.

To understand trust, many look at and try to mimic human trust [4, 5] and consider three main delineations: initial trust, trust evolution, and trust delegation. Initial trust is the first formation of trust between two entities and usually happens through recommendations from other trusted entities. Trust evolution is the continuation and self-adaptation of trust over time and allows for experience to affect the trust relationship. Trust delegation occurs when an entity delegates a trust decision to another trusted entity. In other trust domains [17], different terms express the same basic concepts: experience (for evolution), knowledge (initial), recommendations (delegation). Once we assign a precise meaning and definition for trust, we in essence form a model which may be exercised and evaluated given the assumptions and boundaries of our system. It is essential that regardless of the model chosen, the reason we want to use the model and our expectation of what it will provide must be clearly defined.

2.2 Transitive Trust

We are especially interested in the idea of transitive trust for application to Cybercraft. The goal is to have many Cybercraft working together in the same environment to accomplish a set of goals. There will be times when Cybercraft A must trust Cybercraft B with a trust decision on whether or not to interact with Cybercraft C. In a more general sense, two entities can have varying degrees of trust in each other, within a specific context. An entity can trust an individual in multiple contexts as well, each having a different value of trust. We can express transitive trust as the resulting measure between an entity A and entity C based on the assumption that entity A trusts entity B and entity B trusts entity C. Our interest in transitivity permeates a basic desire for Cybercraft: the ability to take a locality of guaranteed trust (established through hardware) and extend that trust to the execution of code (via payloads) so that the resulting effects, collected data, sensing information, and network operations are trusted as well. The other aspect of trust we must consider deals with the multi-agent communication and cooperation needs that become evident when considering a typical Cybercraft application involving multiple payloads operating across multiple platforms. Both of these application contexts have ramifications for Cybercraft and our desire to express and measure commander-level trust.

2.3 Cybercraft

Phister et al. [15] pose the first conceptual use of Cybercraft as an autonomous, intelligent agent that accomplishes military purposes across a wide variety of electronic-based media. Envisioned as a cyber-vehicle that traverses through cyberspace, Cybercraft were seen as the future platform by which military operations would be conducted in the cyber realm. AFRL enhanced these ideas [2] and developed platform/payload architecture with certain target qualities. The Cybercraft platform, for example, requires a long service life with large investment to support a variety of missions and will be the subject of intense scrutiny to characterize attribution, authentication, and reliability. The Cybercraft payload, likewise, supports rapid development cycles, provides extensibility, and implements specific effects related to defensive missions. The long service life of the platform allows for trust to be formed, maintained, and reevaluated on a constant basis. As research has progressed, trust and a self-protection guarantee for Cybercraft have emerged as a coherent study area [11,14].

We can use a domain model to describe various relationships between entities in a system. Figure 2.1 illustrates a rudimentary domain model for Cybercraft that illustrates several concepts pertinent to trust expression and security. First of all, the platform has a one-to-many relationship with prospective nodes, meaning that one platform will be deployed per node and nodes represent a wide variety of IP-based appliances such as workstations, routers, hand-held devices, or servers. Currently, Cybercraft platforms will execute payloads to achieve or accomplish specific effects and goals in support of operational/tactical missions. Platforms may communicate with other platforms or allow inter-payload communication to accomplish their tasks. Platforms (and thus payloads) may also use other tools or processes (virus checkers, IDS, etc.), depending on the level of trust such tools may have. Though the entire cyber environment is not represented, we can still visualize that nodes are connected to other nodes via networks and are ultimately controlled by some underlying operating system (OS). The specific relationship between the Cybercraft platform and a deployment node is still under consideration, but the current direction assumes a mixture of hardware and firmware that is independent of normal architectural layout.

The tentative domain model in Figure 2.1 reflects the complexity of the trust evaluation process. Payloads are seen to inherit trust from the Cybercraft platform

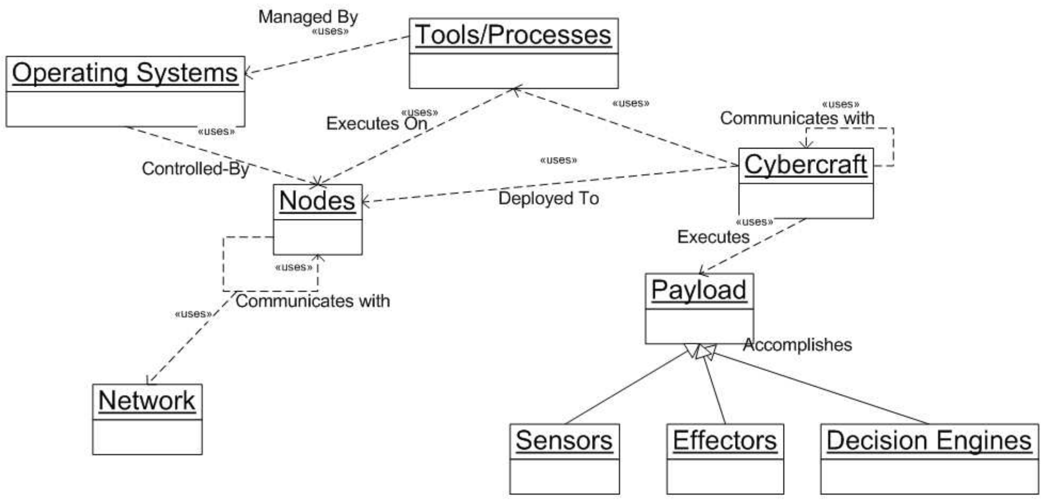

Figure 2.1: Conceptual Domain model for Cybercraft.

and each association represents a possible trust decision that requires evaluation. There will have to be an initial trust value for a Cybercraft to trust the machine it is loaded to. Computer networks are constantly changing and so as time goes on, trust evolution and delegation will need to be addressed. As this domain model is conceptual and by no means as extensive and complex as the Cybercraft domain, it illustrates the need for a way to measure trust between all the interacting parts. Thompson [20] demonstrated through a series of examples that it was impossible to trust any code unless personally written. AFRL has an Anti-Tamper program, Software Protection Initiative (AT/SPI) whose mission is "to prevent the unauthorized distribution and exploitation of application software critical to national security" [1]. Combining these techniques with trust model exploration will only enhance the goal of trusted relationships for the Cybercraft architecture.

2.4 Establishing a Root of Trust

We define a fundamental aspect of trust in Cybercraft as the ability for a system to behave as designed and intended. The notion of a root of trust based on hardware that cannot change is not new - and in fact has been a prized goal for organizations such as the Trusted Computing Group (TCG) for quite some time [21]. TCG specifications provide a starting point for an open set of security related building blocks that will associate trust with all aspects of computing to include storage, networking, software, mobile devices, personal computers, and servers. Two serious concerns for trusted computing standards such as those sponsored by TCG include the notion that competition may be stifled or that manufacturers may implement their "trusted" components incorrectly. In the context of Cybercraft, the former concern is not an issue as the military environment provides the operational bounds and the latter concern would need to be addressed adequately with any proposed Cybercraft platform solution. Nonetheless, the movement towards implementable and procurable secure hardware solutions in the commercial market provides perfect overlap with Cybercraft goals to integrate such technology.

We may liken trust establishment in hardware, software, or even the network itself to the establishment of trust with a service organization. TCG propose the use of silicon-based components such as the Trusted Platform Module (TPM) as a source of trusted storage where keys or passwords may be stored. At a minimum, we require a boot-time process to ensure secure configuration of all further system activity in order for the Cybercraft platform to establish the root of trust. We need to find trust models to capture this Cybercraft aspect and models which exercise further transitive relationships past the platform. Candidate trust models should also address the possibility of physical compromise (capture and subjection of hardware/software to adversarial activity) for either the platform or any possible payload. TCG already distinguishes different roots of trust including measurement, storage, and reporting, which find close corollary to proposed parts of the Cybercraft platform. In the trusted computed realm, we consider attestation as the processes for guaranteeing the accuracy of information and the ability of a platform to vouch for the trustworthiness of another platform. Attestation also provides a parallel notion for a major perceived computing paradigm supported by the Cybercraft architecture involving multi-agent cooperation between payloads accomplishing common tasks and goals.

The root of trust established in hardware for the Cybercraft platform gives us the basis for analyzing transitive trust decisions and gives us a framework to analyze possible trust models. In order to address how we may integrate these models into the development for Cybercraft, we begin with our approach for capturing functional and non-functional requirements, which we discuss next.

2.5 Cybercraft Requirements Distillation

One of the first steps in creating software is defining what it will be used for, in other words, requirements. Current practices for software development include the use of an iterative approach which assumes change and relies heavily on feedback, this is the Object Oriented Analysis and Design (OOA/D) and Iterative Development process. We apply the Iterative Development [12] as a model by which we can perform iterative analysis and design for the Cybercraft architecture while collectively identifying and refining requirements. While some aspects of the future Cybercraft vision may still be in the realm of research and development, the basic requirements for the system derive from a desire to provide comprehensive network defense services in a holistic and secure manner. The requirements for such a system are enormous to say the least and in order to know where to start, we begin with a general understanding of network defense missions as they are currently conducted. However, in order to capture the needs of a future system versus the closed context of current systems, we seek to define the network defense role from a general application perspective. To accomplish this task we use distinct techniques in conjunction: attack/defense trees [6] and use cases [12].

2.5.1 Attack / Defense Trees. If we had free reign to design a holistic approach to network security, based on an extensible architecture that uses mission-specific co-operating payloads, we may best discover the possible tasks of the payloads by looking first at the possible ways our network is attacked. Attack trees provide a textual and visual means to analyze such attacks upon a system and are useful tools to reason about system security. Figure 2 shows an example attack tree where the root node is the attackers goal and the children nodes show means or ways the attacker could accomplish an attack goal. The tree may have AND nodes which mean all child nodes must be successful to achieve the main goal. Trees use OR nodes to represent that only one of its child nodes need to succeed for its path to be successful [6].

Once we exhaustively consider how our networks may be attacked and document those in the form of attack trees, we can then know best what a network defense architecture should be doing in response. For Cybercraft, we apply this approach iteratively by taking specific attacks and creating defense trees in response. Defense trees outline the possible mitigating actions we may take in response. The defense tree corresponds to the leaf nodes of the attack tree and provides us a task' level understanding of what needs to be done. The root node is the attackers goal and the children nodes are means or ways the attacker could accomplish that goal. Figure 2.2 illustrates an example attack tree as dotted lines and in this case shows that a DDoS attack may be defended against using firewall/switch/router ACLs or an intrusion detection system (IDS). Figure 2.3 shows the difference between and AND and an OR node. Though very high level and general in this example, we envision such attack/defense tree modeling will provide a root level of understanding for possible Cybercraft tasks.

Once attack/defense trees are developed, we take some small starting number of trees/branches (our most important roles for example). We then analyze whether those defensive roles are currently being done by a human, an existing tool, or both. In some cases, there may be no (effective) current method that mitigates, detects, or prevents certain attacks. This analysis method gives us a basis to determine whether we want the Cybercraft platform/payload architecture to do a particular task, do a current task better, or possibly automate an existing human-driven process. This process gives us the chance to not only analyze how well (or not) we currently do network defense, but also gives us the ability to look into future requirements without limitation of what currently is possible.

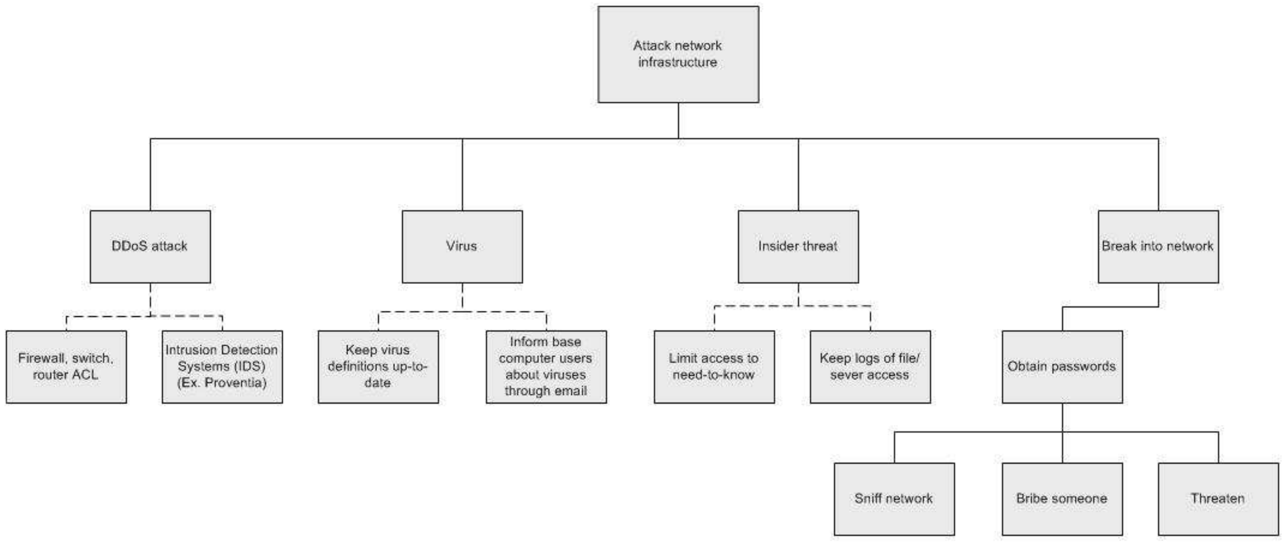
Attack network infrastructure
DDoS attack
Virus
Insider threat
Break into network
Firewall, switch, router ACL
Intrusion Detection Systems (IDS) (Ex. Proventia)
Keep virus definitions up-to-date
Inform base computer users about viruses through email
Limit access to need-to-know
Keep logs of file/ sever access
Obtain passwords
Sniff network
Bribe someone
Threaten

Figure 2.2: Attack tree created for an attack on network infrastructure.

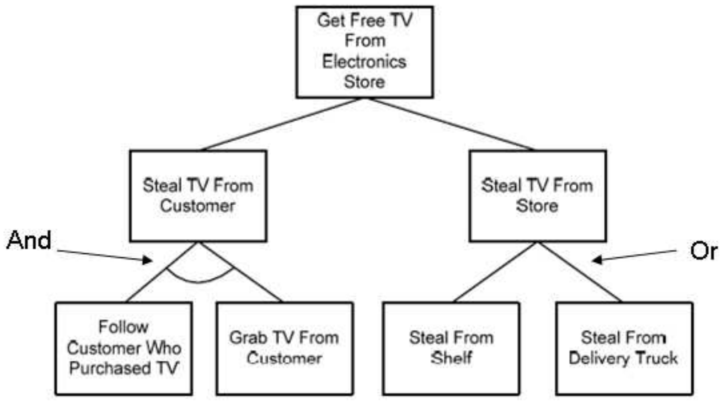
Get Free TV From Electronics Store
Steal TV From Customer
Steal TV From Store
And
Or
Follow Customer Who Purchased TV
Grab TV From Customer
Steal From Shelf
Steal From Delivery Truck

Figure 2.3: Attack tree demonstrating and AND and OR node.

2.5.2 Use Cases. Once we determine defensive roles that are applicable to Cybercraft, we utilize a standard means for capturing software requirements for those roles: use cases. Use cases are textual means of describing the step-by-step interaction between a user and a system. In the case of Cybercraft, we expect that text stories, diagrams, and models will help us determine not only functional requirements for payloads, but also non-functional requirements related to command and control, visualization, and policy development. Because use cases are software methodology agnostic, they provide an ideal means to communicate requirements between users, analysts, and designers. Use cases provide a concrete means to further determine the nature and number of payloads that support a given defensive role or mission. They also provide the ideal means to analyze the impact of transitive trust relationships in a concrete manner.

Not only do Cybercraft requirements need to address how we intend to perform network defense, they also need to uniformly address all possible avenues of vulnerability. We see the nature of the trust question most clearly in this context as it forces us to consider possible ways in which the actual Cybercraft may itself be subverted.

We use scenarios, dependencies, and any implementation assumptions as part of the analysis process to help identify trust expressions (or questions) that should be evaluated or answered. To aid in this process, we also apply the art of misuse cases to the normal use case process. Misuse cases are simply step-by-step descriptions that detail adversarial action and records how the system (should or should not) respond accordingly. We expect this analysis to directly feed our requirements for trust model expression and exercise.

The general template of a use case is shown in Table 2.1. For most of the requirements, we will use brief use cases and thus have only line describing the action.

2.6 Trust Models

In order to address the issue of self protection and trust, we consider the unique aspects of the Cybercraft architecture that need trust model expression and that are revealed as part of the requirements analysis process. We consider several models such as hTrust [5], VTrust [16, 17], and P2P [22]. hTrust , mimics the interactions of humans trust and works well in mobile settings because of the minimal resource demands. VTtrust , is a vector-based trust model. Trust interactions are represented

Table 2.1: Use case template [12].

Use Case Section	**Comment**
Use Case Name	Start with a verb
Scope	The system under design
Level	user-goal or sub-function
Primary Actor	Calls on the system to deliver services
Stakeholders and Interests	Who cares about this use case, and what do they want?
Preconditions	What must be true on start, and worth telling the reader?
Success Guarantee	What must be true on successful completion, and worth telling the reader?
Main Success Scenario	A typical, unconditional happy path scenario of success
Extensions	Alternate scenarios of success or failure
Special Requirements	Related non-functional requirements
Technology and Data Variations List	Varying I/O methods and data formats
Frequency of Occurrence	Influences investigation, testing, and timing of implementation
Miscellaneous	Such as open issues

as relational entities translated to a central database. Peer-to-peer is a trust model applied to peer-to-peer systems.

For each trust model, there are three components of trust: initial trust, trust exchange, and trust evolution. Initial trust is the first formation of trust between two agents. Trust exchange deals with the protocols and exchange of trust between agents. Trust evolution is the continuation of trust over time. This allows for the decay of knowledge that happens over time. Each trust model uses various words for each of these trust ideas but essentially mean the same thing. Table 2.2 shows each model with their terms.

2.6.1 hTrust. hTrust [5] is made up of three main parts: trust formation, trust dissemination, and trust evolution. Trust formation is the initial trust before an interaction occurs; creating a trusting environment that gives us a prediction of

Table 2.2: A summary of similar trust ideas for each trust model

Trust Model	**Initial Trust**	**Trust Exchange**	**Trust Evolution**
hTrust	formation	dissemination	evolution
VTRUST	knowledge	experience	recommended
P2P	ratings generation	ratings discovery	ratings aggregation

trustworthiness. Trust dissemination uses a recommendation exchange protocol to exchange trust opinions. The evolution of trust is the part that allows for a continuous self-adaptation of trust.

2.6.1.1 Trust Formation.

Trust formation is the initial trust before an interaction occurs; creating a trusting environment that gives us a prediction of trustworthiness. To create the initial trust between two agents (say agents a and b), the aggregated trust information, recommendations, and trust formation function Υ are all used.

The aggregated trust information is made up of a set of tuples, as shown in equation 2.1.

$$[a, b, l, s, c, k, t] \tag{2.1}$$

It is read as agent a trusts agent b at level l to do service s in context c. The trust level l is a real value in the range of [-1,1], -1 being total distrust and 1 blind trust. The degree of knowledge k allows for the distinction from unknown and dont trust and ranges from [0,1] with 0 unknown and 1 perfect knowledge. Direct experiences between agents increase the value of k. The last item, t, is a timestamp that allows for the fact that knowledge decays with time.

Recommendations follow the same format as the aggregated trusted information tuple with the addition of the recommenders private key. They are used to form initial trust opinions and delegation, such as relying on third-party assessments. The format is shown in equation 2.2

$$[a, b, l, k, t]_{SK_x} \tag{2.2}$$

The trust formation function Υ will return a predicted trust value or range to base trust opinions upon and is used to predict the trustworthiness of a trustee.

2.6.1.2 Trust Dissemination. Trust dissemination uses a recommendation exchange protocol to exchange trust opinions. Each agent carries a portfolio of credentials in their local environment. A portfolio consists of a set of letters representing the history of the agent. The letter is a tuple, following the same format as recommendations. Agent a will form a trust opinion about agent b by using the protocol. Below are the steps.

1. Agent a sends agent b a request for b's portfolio of credentials
2. Agent b replies with a set of at most m letters of presentation
3. Trust formation function Υ forms a trust opinion about agent b based on the information from the letters
4. Interaction between agents a and b may or may not take place depending on the resulting trust value from trust formation function Υ

2.6.1.3 Trust Evolution. The evolution of trust is a fundamental concept of any TMF and allows for a continuous self-adaptation of trust. The customizing functions used for trust evolution are the aggregated function Φ and the tacit information extraction function Ψ . Trust evolution also plays a role in catching malicious agents.

The aggregation function Φ maintains information about the trustworthiness of an agent as a service provider and is used to update the perceived trustworthiness of trustee b when new direct experiences occur. The extraction function Ψ maintains information about the trustworthiness of an agent as a recommender and is the subjective part of the TMF. Recommendations are given different weight values using the extraction function and are based on how much trust we have in the agent the recommendation came from.

Malicious agents will send bad information, spreading fake good recommendations and fake bad recommendations. To help guard against this, a boundary trustworthiness value $\eta \in$ [-1,1] is defined for each agent so that when an agents trustworthiness drops below η, the agent becomes suspect. When an agent becomes suspect, all recommendations coming from that agent are discarded.

2.6.2 VTrust. The trust relationship in VTrust [16, 17] consists of a vector with three components: experience, knowledge, and recommendation. Experience is the number of events two agents share within a certain timeframe. Knowledge is composed of direct knowledge and indirect knowledge. A recommendation uses a recommendation value as its basis for trust. As well, neutrality is acceptable in this model. Stevens [19] considers the application of trust vectors and their applicability for Cybercraft fitness. Their analysis examines the use of Cybercraft payloads in multi-agent information gathering roles where agents may have to evaluate information from other agents. Such roles cover a large number of defense applications where network sensing data are analyzed. Their conclusions show that a modified Trust Vector model could meet the needs for expressing trust decay and transitive trust decisions in the information retrieval context.

The trust relationship is calculated from three numeric values represented as a decimal ranging from [-1, 1] $\cup \perp$. A negative value represents trust-negative, a positive value represents trust-positive and a zero value is trust-neutral. Lack of value from insufficient data is given by $\perp$. This trust relationship is shown in equation 2.3

$$(A \rightarrow^{c} B)_t = [{}_A E^{c}_{B}, {}_A K^{c}_{B}, {}_{\psi} R^{c}_{B}] \tag{2.3}$$

Experience is the number of events two agents share within a certain timeframe. An event can be trust positive, trust negative, or trust neutral. Recent events are given more weight.

Knowledge is composed of direct knowledge and indirect knowledge. Each component of the trust model (experience, knowledge, and recommendation) is given two values; one to represent direct and the other indirect knowledge. The knowledge policy computes the weight these values are given. Knowledge is gathered by summing of the product of the two knowledge values.

A recommendation uses a recommendation value as its basis for trust. An agent uses the level of trust ([-1,1]) to provide a weighed recommendation. This weight multiplied with the former value will return the recommendation score for an agent.

The VTrust model allows for two agents coming up with different trust values from the same input. This can happen when an agent gives different weight values for each trust component (experience, knowledge, recommendation).

2.6.3 P2P. In peer-to-peer (P2P) systems, peers interact with unknown peers without trusted third-parties. In P2P systems there is a frequent join and leave of peers. Yu, et al. [22] present a distributed approach for P2P systems using reputation mechanism for trust. Their approach uses polling algorithms and deals with dishonest or unlreliable peers. Peers can be thought of as agents (from the previous models discussed). There are three reputation mechanisms: ratings generation, ratings discovery, and ratings aggregation. Each are discussed below.

2.6.3.1 Ratings Generation. Ratings generation focuses on how to aggregate ratings and is represented an interval from [0,1]. There are two different types of ratings: service (reliability) and voting (credibility). Tying these words into the other models, service is a service specific trust opinion and voting is a recommendation trust opinion. A peer P_i wanting to evaluate the trustworthiness of peer P_j has two options: using direct experience and recommendations. Direct experience is interaction between the two peers and recommendations come from others in the case where P_i and P_j have no frequent interactions.

The two different types of ratings are local rating and aggregate rating. Local rating is based on direct interaction and is generated every time an interaction takes place between two peers, such as P_i and P_j. Aggregate rating combines the local ratings (if there are any) with recommendations from other witnesses. This is used to decide whether peer P_j is trustworthy and whether peer P_i will propagate the ratings to others.

Reputation mechanisms, or aggregate ratings, help establish trust between peers but direct interaction, or local rating, has more weight in the decision for P_i to trust P_j. In the case that P_i and P_j have not interacted (no local ratings), if enough recommendations have been received, interaction will occur.

2.6.3.2 Ratings Discovery. Ratings discovery uses the process of referrals to find witnesses in an efficient manner. If a witness is found, the response is sent in the reverse path of the request. A series of referrals makes a referral chain. This can be thought of as transitive trust or chain trust, one of the key ides for Cybercraft. The referral chain creates a trust graph that is kept in the peers' local environment.

2.6.3.3 Ratings Aggregation. There needs to be a way to distinguish between reliable and deceptive or unreliable peers. Witnesses may not give true information about other peers. This is called noisy ratings and has three variations: complementary, exaggerated positive, and exaggerated negative. Malicious peers are either individual or in a colluding group. Weighted majority techniques are used to predict the trustworthiness of a given party. A peer will maintain a weight for the credibility of each peer it requests a testimony from that gives an estimate of how credible the peer thinks the witness is.

2.7 Summary

Our goal for this research is to further the process of requirements gathering using the tools of attack/defense trees and use/misuse cases. We discussed in this chapter the basic ideas of trust, along with the need for transitive trust representation for Cybercraft. Then we discussed the Cybercraft domain and what that entails thusfar. Finally, we gave an overview of three trust models that have the potential to be applied to Cybercraft to give a value to trust relationships. The next chapter uses the tools to generate requirements for Cybercraft and Chapter IV goes over the three trust models in greater detail.

III. Cybercraft Requirements Development

Requirements' gathering is a daunting, but needed task, especially for the ideas and goals for Cybercraft. As mentioned in Chapter II, use cases and attack and defense trees are used to flesh out some functional and non-functional requirements. Using a strategic-operational-tactical view, we create probable Cybercraft missions and/or payloads from each of the defense priorities. This chapter is laid out as follows. The next section starts the process for hashing out requirements, followed by a couple of examples of expanded use cases, and finally the chapter closes with a summary of Cybercraft requirements.

3.1 Requirements gathering

Using the top ten defensive priorities[1], we compile a list of brief use cases using a strategic-operational-tactical mission level breakdown process. The defense priorities expanded on are: attack detection, automated network vulnerability mitigation, automated attack interdiction, network attack damage assessment, automated attack reporting, and adversary identification [2]. Use cases were created for each of these defense priorities using a mision level task breakdown and are shown in Appendix A.

3.1.1 Attack Detection. The first defense priority is attack detection. The ability to detect an attack before, during, or after, is crucial to keeping our networks secure. USAF networks are constantly under attack and our network defenders must be able to respond quickly.

Many items from can be implemented into a Cybercraft mission. Creating logs, maintaining password policies, monitoring IP addresses, and monitoring ports are all good candidates to become Cybercraft payloads or missions. The other items listed could possibly still be done with Cybercraft payloads, but would need more of a breakdown and thought put into them.

[1]Air Force ACC IO RAWG 2006

[2]This work was accomplished with the help of Mr. Lou Giannelli, Contractor, USAF ACC 83 NOS Det 3/SCN, Lead Network Defender

With these use cases, we can create numerous attack/defense trees. Taking a specific attack, a Code Red buffer overflow, we create a defense tree shown in Figure 3.1. The corresponding attack tree is shown in Figure 3.2. A Code Red buffer overflow attack attempts to connect to TCP port 80, and once connected, sends a GET request to exploit a buffer overflow [3].

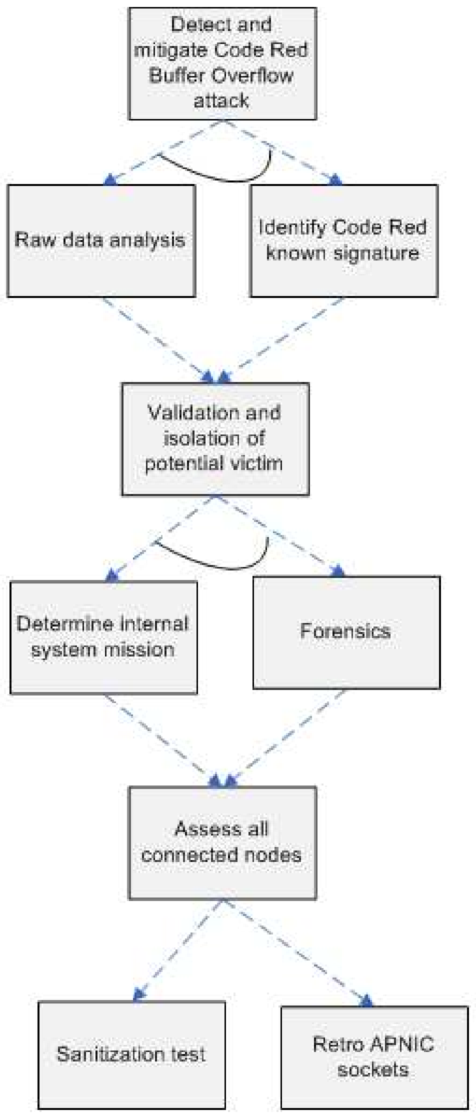
Detect and mitigate Code Red Buffer Overflow attack
Raw data analysis
Identify Code Red known signature
Validation and isolation of potential victim
Determine internal system mission
Forensics
Assess all connected nodes
Sanitization test
Retro APNIC sockets

Figure 3.1: A defense tree for detecting a Code Red buffer overflow attack.

[3]retrieved from http://www.cert.org/advisories/CA-2001-19.html

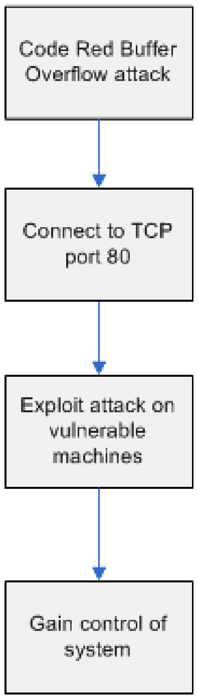
Code Red Buffer Overflow attack
Connect to TCP port 80
Exploit attack on vulnerable machines
Gain control of system

Figure 3.2: An attack tree for a Code Red buffer overflow attack.

A defense tree is useful in determining what is being done now with current tools, processes, or by human. Currently, traffic validation is done by tools and by network defenders. There is no way to specify a list for a Cybercraft to look for all the vulnerabilities and many are found by human analysis and piecing things together. However, a Cybercraft could be tasked to isolate a specific network or workstation. This could enhance the speed with which vulnerabilities can be exploited and the window of vulnerability is drastically reduced. Finally, it is possible for a Cybercraft to implement forensic guidelines if they were scrutinized.

Referencing the Code Red attack tree, there are certain generalizations. First, a Cybercraft could dispatch a payload with AV software to detect these types of known attacks. As well, Cybercraft could be more dynamic and react faster than AV software if it is always running. AV software is supposed to detect known signatures of attacks, but only when it is running. A Cybercraft could be placed specifically to detect these types of attacks 24/7.

3.1.2 Automated Network Vulnerability Mitigation. The second defense priority we use is automated network vulnerability mitigation. Although it is impossible to automate human analysis, we can try to automate tasks that are repetitive and require searching for specific known items.

Anything that requires monitoring or collecting data can be automated and are potential for Cybercraft payloads. A payload could be used to parse data for specific items of "interest". These items would have to be enumerated by network defender SME's (subject matter expert). A point needs to be maide here that it is difficult to automate the analysis of collected data. The ability of human analysis and interaction cannot be replaced by any tool, no matter what it boasts.

Checking vulnerabilities is a good area for Cybercraft. A baseline of parameters can be set and a Cybercraft can notify the operator when a change occurrs. An example is ensuring current USAF network policies are followed. Enforcing these policies is another possible Cybercraft mission. TCNO's (Time Compliance Network Order), TCTO's (Time Compliance Technical Order), and AFI's (Air Force Instructions) contain many of these policies and procedures and can also be used to hash out possible Cybercraft missions and payloads. The specifics of these documents are beyond the scope of this thesis and will be published as an area of future research.

Automated patching is currently being done on USAF networks with Microsoft© SMS (System Management Server). There are many issues with the implementation of SMS on USAF networks. Because of the varied networks and programs that are mission essential that need specific OS's there is often a large exemption list. An exemption list includes workstations and possibly even entire networks exempt from the automatic pushing of patches from SMS. This exemption list creates numerous vulnerabilities on the network. This is where Cybercraft can come in. It has the potential to create new software or enhance the old not so functional software to actually be beneficial and help ensure security for the network.

3.1.3 Automated Attack Interdiction. The next defense priority we use is automated attack interdiction. We can't stop enemies from trying to attack our networks, but we can work to interdict and reflect the attacks. The use cases created for this defense priority are shown in Appendix A.3.

If a device becomes suspect, a Cybercraft will be able to isolate the device in a more timely and efficient manner and mitigate further devices from becoming suspect than current tools and processes. This greatly enhances security for our networks as we can respond at the speed of the network instead of at human speed. We envision Cybercraft to be an autonomous agent, and thus be able to adapt to what is happening to the network.

The corresponding attack and defense trees created for SQL inject attack are shown in Figures 3.3 and 3.4 respectively. Figure 3.3, the defense tree, shows a general sequence of events for detecting and mitigating a SQL injection attack. This particular case is mainly done with analysis from network defenders. A possible Cybercraft mission for this scenario is blocking the IP segments. A Cybercraft could accomplish this much faster than the human process.

The attack tree, Figure 3.4, steps through a possible sequence an attacker could use to accomplish a SQL inject attack. The main goal if this attack is to gain access to unauthorized information by taking advantage of unfiltered user input. To protect against these attacks, user input must be filtered and always checked before blindly being used by the program.

3.1.4 Network Attack Damage Assessment. The next defense priority we use is network attack damage assessment. It's important to know the who, what, when, where, and why of an attack, or at least as much information as you can. As

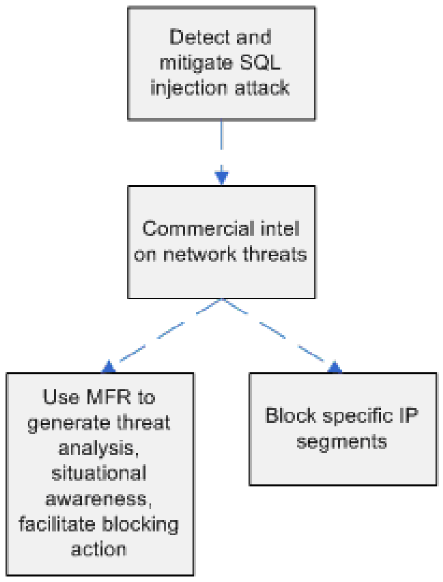
Detect and mitigate SQL injection attack
Commercial intel on network threats
Use MFR to generate threat analysis, situational awareness, facilitate blocking action
Block specific IP segments

Figure 3.3: A defense tree for detecting a SQL Inject attack.

the old saying goes, keep your enemies closer than your friends. The use cases created for this defense priority are shown in Appendix A.4.

Standard desktop is a goal for USAF leaders that all workstations have the same look and feel to them. At first glance, this would make security easier, but there are always exceptions and keeping up with exceptions or an exemption list is always hard to do. An exemption list means manually updating and patching. Since one of the goals is to have a Cybercraft installed on every machine, it could take control of keeping the computer up-to-date.

Currently, the HP Openview suite of network management products is used to monitor and report on the health of the network. We do not envision Cybercraft to take over all the responsibilities of network defense, as there are plenty of current software/hardware solutions already in place (such as HP Openview). The goal of Cybercraft is to supplement these current tools to enhance network security and ensure we are protecting ourselves adequately.

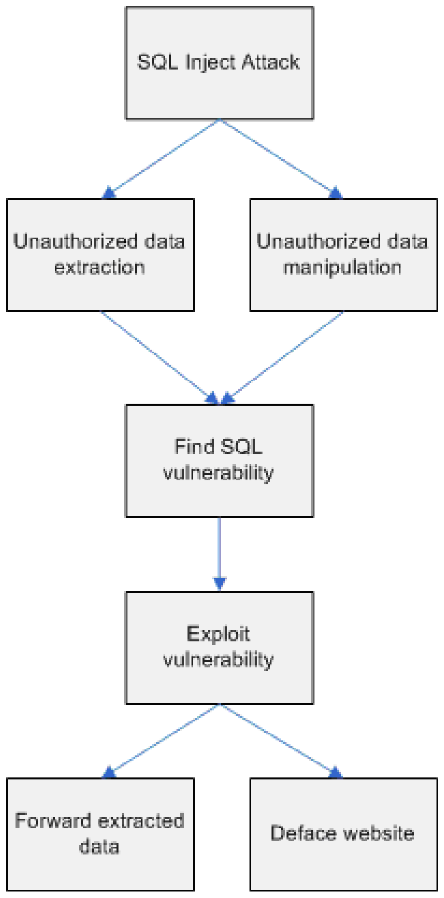

Figure 3.4: An attack tree for a SQL Intect attack.

3.1.5 Automated Attack Reporting. The next defense priority we use is automated attack reporting. As well as knowing the who and why, it's important to keep track of items for future analysis to detect patterns. The use cases created for this defense priority are shown in Appendix A.5.

Tracking incidents gives us documentation to compare and find patterns an enemy is using. It will be difficult to place a specific role to Cybercraft for this use case as this is mostly a network defender analysis role. If there is anything to automate, Cybercraft would be a prime candidate.

3.1.6 Adversary Identification. The last defense priority we use is adversary identification. Going along with the last two defense priorities, knowing who your enemies are is extremely important. The use cases created for this defense priority are shown in Table A.6.

These use cases go hand-in-hand with the previous section. Along with anlayzing data and detecting patterns, finding out the source of the attacks is very important. Knowing what your enemy is up to can help predict a future attack and thus help us prepare for the future predicted event far better than not knowing anything in advance.

3.2 Expanded Use Cases

Our first scenario deals with anti-virus (AV) software and ensuring proper configuration on all workstations on a network. We go through the scenario in Table 3.1. Our next scenario deals with the firewall software and ensuring proper configuration. We go through the scenario in Table 3.2. These are only two examples of hashing out details and figuring out where we can use Cybercraft and deploy their payloads. Ideally, we would create fully-dressed use cases for all the brief use cases mentioned in this chapter. This is an area of future work.

3.3 Summary of Cybercraft Requirements

It is important to enumerate requirements of Cybercraft. We need to know what we are expecting them to do before trying to implement anything. Below is a summary of suggestions.

- Ensure secure condition on the node using integrity check
- Supplement SMS
- Port monitoring

Table 3.1: Use case for ensuring AV is installed and up-to-date.

Use Case Name	AV
Scope	The network
Level	Ensure AV software is installed and up-to-date on all machines
Primary Actor	Cybercraft
Stakeholders and Interests	Network Defenders
Preconditions	Network is operational, up-to-date,
Success Guarantee	All machines have AV software loaded, operational, and up-to-date
Main Success Scenario	Cybercraft platform creates a payload to check AV software on all machines in the network, if all machines have operational AV that is up-to-date, the scenario is successful
Extensions	Cybercraft platform creates a payload to check AV software on all machines in the network. Alternate scenarios: 1. If there is no AV software, the Cybercraft platform dispatches another payload to install AV software on the machine in question 2. If there is AV software installed, but not updated, the Cybercraft platform dispatches another payload to obtain correct updates from approved sites
Frequency of Occurrence	Daily
Miscellaneous	Assumptions are that the Cybercraft payload and platforms are trusted, the network is secure, all channels a Cybercraft uses are secure

- Network monitoring
- Anomaly detection, send an alert if out of the ordinary
- Flagging an alert on outbound traffic when certain conditions are met
- Enforce TCNO/TCTO, AFI network operating procedures
- Parse raw data transcripts to help base technicians locate key elements [4]

[4]For more information on this bullet, refer to Appendix B

Table 3.2: Use case for ensuring firewall is installed and up-to-date.

Use Case Name	Firewall
Scope	The network
Level	Ensure firewall software is installed and up-to-date
Primary Actor	Cybercraft
Stakeholders and Interests	Network Defenders
Preconditions	Network is operational, up-to-date,
Success Guarantee	All boundary devices have a firewall that is operational, and up-to-date
Main Success Scenario	Cybercraft platform creates a payload to ensure firewall is installed. If up-to-date and configured properly, the scenario is successful
Extensions	Cybercraft platform creates a payload to check the status of the firewall. Alternate scenarios: 1. If there is no firewall, the Cybercraft platform dispatches another payload to install a new firewall 2. If there is a firewall installed, but not updated, the Cybercraft platform dispatches another payload to obtain correct updates from approved sites and install them, if any
Frequency of Occurrence	Daily
Miscellaneous	Assumptions are that the Cybercraft payload and platforms are trusted, the network is secure, all channels a Cybercraft uses are secure

IV. Trust Model Setup

For Cybercraft, we need a model to capture not only the subjectability of trust, but the decision making, learning, and obeying aspects as well [7]. The decision making process allows for trust delegation and when this should occur. Cybercraft must know who is friendly to be able to interact and learn information from and be able to delegate all authority and obey another with confidence. This chapter explains each trust model in greater detail, examining the underlying mathematics that make the models work.

4.1 hTrust

hTrust has three trust values: trust formation, trust dissemination, and trust evolution (reference Chapter II). Stevens states [18] all Cybercraft platforms will go through a formal verification but not all payloads will. Therefore, one can conclude that certain payloads will have vastly different trust values. For reference, trust information is kept as a set of tuples, shown below in Equation 4.1.

$$[a, b, l, s, c, k, t] \tag{4.1}$$

The minimized tuple shown in Equation 4.2 is referenced henceforth and is read agent a trusts agent b at level l, knowledge k at time t. l represents the level agent a has in agent b and is represented as a range from [-1,1]. k is the degree of knowledge, in other words, how much agent a knows about agent b, and is represented as a range from [0,1].

$$[a, b, l, k, t] \tag{4.2}$$

Trust formation and trust dissemination use trust formation function Υ. The trust evolution phase uses the two other functions: the aggregation function Φ and the tacit information extraction function Ψ.

4.1.1 Trust Formation Function Υ. Equation 4.3 is the mapping of Equation 4.4. Υ_{op} is a trust prediction (range from [-1,1]) based on one trust opinion (or tuple). $\mathcal{O}$ is the set of all trust opinions whereas ε is the set of all environments. Thus, Υ_{op} , shown in Equation 4.3 is read as the set of tuples in $\mathcal{O}$ map to the environment ε, in the range [-1,1].

Equation 4.4 calculates the trust range using l, k, and t from a tuple. T and η are parameters from the agents local environment. T is the time interval trust information is gathered and η is a boundary value and agents aren't trusted if their trust opinions fall below η. Equation 4.4 is run on each tuple to achieve a trust value range.

$$\Upsilon_{op} : \mathcal{O} \rightarrow \varepsilon \rightarrow [-1,1] \times [-1,1] \tag{4.3}$$

$$\Upsilon_{op}[a,b,l,k,t]_e = [max(-1, l- \mid l - f \mid), min(l+ \mid l - f \mid, 1)] \tag{4.4}$$

$$f = l * k * max\left(0, \frac{T - (t_{now} - t)}{T}\right) \tag{4.5}$$

Another range is calculated (Equation 4.7) from a set of m recommendations received. Equations 4.8, 4.9 and 4.10 are used in Equation 4.7.

$$\Upsilon_{rec} : \wp(\mathcal{O}) \rightarrow \varepsilon \rightarrow [-1,1] \times [-1,1] \tag{4.6}$$

$$\Upsilon_{rec}[\{o_i \mid i \in [1,m]\}]_e = [l_{low}, l_{high}] \tag{4.7}$$

$$l_{low} = \frac{\sum_i \{\pi_1(\Upsilon_{op}[o_i]_e) * q_i \mid q_i > \eta\}}{\sum_i (q_i \mid q_i > \eta)}, \quad l_{high} = \frac{\sum_i \{\pi_2(\Upsilon_{op}[o_i]_e) * q_i \mid q_i > \eta\}}{\sum_i (q_i \mid q_i > \eta)} \tag{4.8}$$

$$\pi_1(l_1, l_2) = l_1, \quad \pi_2(l_1, l_2) = l_2 \tag{4.9}$$

$$q_i = max(\eta, l_i^{'} * k_i^{'} * max\left(0, \frac{T - (t_{now} - t_i^{'})}{T}\right)) \tag{4.10}$$

Equation 4.11 is the mapping of trust formation function Υ. The set of all tuples $\mathcal{O}$ crossed with the set of all recommendations $\wp(\mathcal{O})$ maps to the environment ε which produces the result of a trust value range from [-1,1] $\times$ [-1,1].

$$\Upsilon : \mathcal{O} \times \wp(\mathcal{O}) \rightarrow \varepsilon \rightarrow [-1,1] \times [-1,1] \tag{4.11}$$

$$\Upsilon[(o, \mathcal{O})]_e = h_1(\Upsilon_{op}[(o)]_e, \Upsilon_{rec}[\mathcal{O}]_e) \tag{4.12}$$

The customizing function, h_1 (Equation 4.13), is applied to the range produced from the function Υ and the final result is a prediction of trustworthiness between two agents a and b. The application chooses a value from this range to use as a trust prediction.

$$\begin{aligned} h_1 = ([l_1, l_2], [l_1^{'}, l_2^{'}]) = &[h_2([l_1, l_2]) - \mid h_2([l_1, l_2]) - h_2([l_1^{'}, l_2^{'}]) \mid, \\ &h_2([l_1, l_2]) + \mid h_2([l_1, l_2]) - h_2([l_1^{'}, l_2^{'}]) \mid] \end{aligned} \tag{4.13}$$

$$h_2(l_1, l_2) = w_1 * l_1 + w_2 * l_2 \tag{4.14}$$

4.1.2 Aggregation Function Φ. The aggregation function Φ is used to update an agent's local environment when a new direct experience occurs. It is composed of three equations. The first equation (4.15), shows the mapping. A range from [-1,1] crossed with the set of all recommendations maps to the environment.

$$\Phi : [-1,1] \times \wp(\mathcal{O}) \rightarrow \varepsilon \tag{4.15}$$

Equation 4.16 is the first of two formal definitions for aggregated function Φ. It describes the case where agent a's local environment is updated from a new direct experience with agent b. Customizing function, h_4 (Equation 4.19), combines the three trust opinions (agent a's old trust opinion about agent b, the trust opinion just received about agent b from the new experience, and the newly updated trust opinion using the new recommendations) to create the new trust opinion a will carry about b in its local environment.

Weights are used in Equation 4.19 to weight the different values of l. l_1 is the trustworthiness of b from the recent interaction, l_2 is the opinion previously held by a and l_3 is b's expected trust value based on the recent experience. Depending on which one is valued over the other will determine the value of each weight. For example, if I have a friend for many years whom I place a lot of trust in and a recent experience with her is not good, her previous experience with me will weigh more than the most recent experience. Tying this example to Equation 4.19, $w_1 = 1$, $w_2 = 0$, and $w_3 = 0$ to say that I trust past experiences more than most recent and perceived from the most recent. But, if I only recently met this same friend, the most recent experience is going to weigh more because there isn't much past data to compare against. Thus, $w_1 = 1$, $w_2 = 1$, and $w_3 = 0$ which will change trust opinions fairly quickly because we are taking into account two of the three trust opinions.

$$\begin{gathered}\Phi[\tilde{l}, O]_e = e \setminus \{[a,b,l,k,t]\} \cup \{[a,b,l^{'},k^{'},t^{'}]\} \mid o = lookup(b,e) = [a,b,l,k,t] \\ \wedge\, l^{'} = h_4(\tilde{l}, l, h_2(\Upsilon_{rec}[O]_e)) \wedge k^{'} = min(k + k_{min}, 1) \wedge t^{'} = t_{now}\end{gathered} \tag{4.16}$$

The second definition of aggregated function Φ is Equation 4.17. Equation 4.17 considers the case where a trust opinion updates soley based on recommendations and no interaction between agents a and b occured. This trust opinion is calculated from

l (old trust opinion from a) and $h_2(\Upsilon_{rec}[O]_e)$. k is not updated as knowledge is only received from direct experiences [5].

$$\begin{aligned}\Phi[\varepsilon, O]_e = \ & e \setminus \{[a,b,l,k,t]\} \cup \{[a,b,l^{'},k^{'},t^{'}]\} \mid o = lookup(b,e) = [a,b,l,k,t] \wedge \\ & l^{'} = h_4(\varepsilon, l, h_2(\Upsilon_{rec}[O]_e)) \wedge k^{'} = k \wedge t^{'} = max(t, max(\{\pi_{time}(o_i), o_i \in O\}))\end{aligned} \tag{4.17}$$

$$h_3 : [-1,1] \times [-1,1] \rightarrow \{-1,1\} \tag{4.18}$$

$$h_4(l_1,l_2,l_3) = \frac{w_1 * l_1 + w_2 * l_2 + w_3 * l_3}{w_1 + w_2 + w_3} \tag{4.19}$$

4.1.3 Tacit Information Extraction Function Ψ. The tacit information extraction function Ψ is used to weigh recommendations differently when an agent must make a trust decision with no previous direct experiences and can only rely on recommendations. For example, agent a might have a higher trust value for agent b than agent c and thus agent b will have a higher weight value placed on his recommendations.

Equation 4.15 is the mapping of tacit information extraction function. A range from [-1,1] crossed with the set of all recommendations maps to the environment and the new value then maps to the environment.

$$\Psi : [-1,1] \times \wp(\mathcal{O}) \rightarrow \varepsilon \rightarrow \varepsilon \tag{4.20}$$

The Trust Management Framework (TMF) maintains a set of tuples for each agent as a recommender, which is referred to as tacit information. The tacit information extraction function Ψ updates this information after an interaction using

[5] If the knowledge component is able to be transitively updated in this equation, chain trust will work for this model. Reference the results in Chapter VI

Equation 4.21. The tacit information tuple is updated based upon the perceived trustworthiness of a recent interaction between agents a and b and the recommendation about b from a recommender.

$$\begin{aligned}
\Psi[\tilde{l}, O]_e = \ & e \setminus \{r_i = lookup(i, e) = [a, i, l_i, k_i, t_i], \forall o_i \in O\} \cup \{r_i^{'} = [a, i, l_i^{'}, k_i^{'}, t_i], \forall o_i \in O\} \mid \\
& k_i^{'} = min(k_i + k_{min}, 1) \wedge l_i^{'} = \begin{cases} max(-1, h_5(l_i, \delta l_i)) & \text{if } \delta l_i > \delta_{max} \\ min(h_5(l_i, \delta l_i), 1) & \text{if } \delta l_i \leq \delta_{max} \end{cases}, \\
& \delta l_i = \mid l^{'} - h_2(l_i - \mid \pi_l(o_i) - \pi_l(o_i) * \frac{T - (t_{now} - t_i)}{T} \mid, \\
& \qquad l_i + \mid \pi_l(o_i) - \pi_l(o_i) * \frac{T - (t_{now} - t_i)}{T} \mid) \mid
\end{aligned} \tag{4.21}$$

The customizing Function h_5 (4.22), creates a new trust value $l_i^{'}$ based on the agents past trustworthiness and a discrepancy. If the discrepancy of opinions is lower than the tolerance parameter, then trustworthiness is increased. If the discrepancy is higher, the trustworthiness is decreased. Another possibility is weighing the past and the new recommendation equally, thus the last equation (where n is not a factor at all). The tolerance parameter decides what an agent will select for a recommender. The lower the tolerance, the more restrictive the agent is in selecting recommenders.

$$\begin{aligned}
h_5: \quad l_i = \frac{n * l_i + \frac{2 - \delta l_i}{2}}{n + 1}, \quad OR \quad l_i = \frac{n * l_i - \frac{2 - \delta l_i}{2}}{n + 1}, \\
OR \quad l_i = \frac{l_i + \frac{2 - \delta l_i}{2}}{n + 1}, \quad where \quad n = \frac{k_i}{k_{min}}
\end{aligned} \tag{4.22}$$

Table 4.1 is an agents local environment. T represents the time interval when interactions are observed. t_{now} is the current time the trust value is calculated. t is the trust value from the tuple being used to calculate the trust opinion and η is a boundary value in the range of [-1,1] that represents the cutoff of whether an agent trusts another agent below a certain value.

Table 4.1: Agent $a's$ Local Environment [5].

Data	
Aggregated Trust Information	$[a, x, l, k, t]$
Tacit Information	$[a, x, l, k, t]$
Portfolio of Credentials	$[x, a, l, k, t]_{SK_x}$
Parameters	
Time Interval of Relevant Observations	T
Maximum Tolerate Discrepancy of Opinions	δ_{max}
Single Increment of Knowledge	k_{min}
Minimum Trust Level	η
Customizing Functions	
Given two trust ranges, compute a trust range (used by Υ)	h_1
Given a trust range, compute a trust opinion (used by Υ)	h_2
Given a trust range, decide whether the prediction is precise enough (used by the recommendations exchange protocol)	h_3
Given three trust opinions, compute a new one (used by Φ)	h_4
Given a trust opinion and a discrepancy, compute a new trust opinion (used by Ψ)	h_5

4.2 VTrust

VTrust is composed of three components: experience, knowledge, and recommendation. Equation 4.23 represents the vector. ${}_AE^c_B$ is the magnitude of A's experience about B in context c, ${}_AK^c_B$ is A's knowledge and ${}_\Psi R^c_B$ is the affect of B's recommendations to A. Each of these three values fall in a range from [-1,1] $\cup \perp$ where no knowledge is represented by $\perp$.

$$(A \rightarrow B)_t = [{}_AE^c_B, {}_AK^c_B, {}_\Psi R^c_B] \tag{4.23}$$

4.2.1 Experience. Experience is the number of events between two agents A and B within a specific time frame $[t_0, t_n]$. Steven's work [18, 19] concluded that keeping between 9 to 10 intervals is sufficient for Cybercraft, depending on storage and how much granularity is needed. Equation 4.24 gives the value for experience . w_i is a non-negative weight and I_i is the sum of all values of events (trust-positive, trust-negative, trust-neutral).

$$ {}_{A}E^{c}_{B} = \frac{\sum_{i=1}^{n} w_i I_i}{\sum_{i=1}^{n} n_i} \tag{4.24}$$

$$ w_i = \frac{i}{S}, where\ S = \frac{n(n+1)}{2} \tag{4.25}$$

4.2.2 Knowledge. The knowledge component is composed of two parts: direct knowledge and indirect knowledge. Equation 4.26 gives the knowledge component. d and r are in the range [-1,1] $\cup\{\perp\}$, represent direct and indirect knowledge, respectively, and $w_d + w_r = 1$.

$$ {}_{A}K^{c}_{B} = \begin{cases} d, & if r = \perp \\ r, & if d = \perp \\ w_d \cdot d + w_r \cdot r, & if d \neq \perp, r \neq \perp \\ \perp, & if d = r = \perp \end{cases} \tag{4.26}$$

4.2.3 Recommendation. The recommendation is calculated using a recommendation value and the level of trust the agent has in the recommender. The recommendation value represents the level of trust an agent A has for a recommender agent B. The equation for a recommendation is 4.27.

$$ {}_{\Psi}R^{c}_{B} = \frac{\sum_{j=1}^{n} (v(A \rightarrow j)^{N}_{t}) \cdot V_j}{\sum_{j=1}^{n} (v(A \rightarrow j)^{N}_{t})} \tag{4.27}$$

Ψ is a group of n recommenders and $v(A \rightarrow j)^{N}_{t}$ is a trust value from the j^{th} recommender and V_j is the recommender's recommendation value.

4.2.4 Normalization Policy. The normalization policy takes into affect the different weights a trustor may assign during trust evaluation. It is represented as a vector and the breakdown is displayed in Equation 4.28. Trust changes and decays with time and thus Equation 4.29 represents the time affected vector.

$$\begin{aligned}(A \to B)_t^N &= \mathbf{W} \odot (A \to B)_t \\ &= [W_e, W_k, W_r] \odot [{}_A E_B^c, {}_A K_B^c, {}_\Psi R_B^c] \\ &= [W_e \cdot {}_A E_B^c, W_k \cdot {}_A K_B^c, W_r \cdot {}_\Psi R_B^c] \\ &= [{}_A \hat{E}_B^c, {}_A \hat{K}_B^c, {}_\Psi \hat{R}_B^c]\end{aligned} \tag{4.28}$$

$$(A \to B)_{t_n}^N = \begin{cases} [{}_A\hat{E}_B^c, {}_A\hat{K}_B^c, {}_\Psi\hat{R}_B^c] & if\ t_n = 0 \\ [\frac{v(\hat{T})}{3}, \frac{v(\hat{T})}{3}, \frac{v(\hat{T})}{3}] & if\ t_n \neq 0\ and\ {}_A\hat{E}_B^c = {}_A\hat{K}_B^c = {}_\Psi\hat{R}_B^c = \perp \\ \alpha \cdot [{}_A\hat{E}_B^c, {}_A\hat{K}_B^c, {}_\Psi\hat{R}_B^c] + \beta \cdot [\frac{v(\hat{T})}{3}, \frac{v(\hat{T})}{3}, \frac{v(\hat{T})}{3}] & \\ \quad if\ t_n \neq 0\ and\ at\ least\ one\ of & \\ \quad\quad {}_A\hat{E}_B^c, {}_A\hat{K}_B^c, {}_\Psi\hat{R}_B^c \neq \perp & \end{cases} \tag{4.29}$$

Trust values are calculated using Equation 4.30.

$$\mathbf{v}(A \to B)_t^N = {}_A\hat{E}_B^c + {}_A\hat{K}_B^c + {}_\Psi\hat{R}_B^c \tag{4.30}$$

4.3 P2P

P2P's three main components are: ratings generation, ratings discovery, and ratings aggregation.

4.3.1 Ratings Generation. The two types of ratings are local and aggregate ratings. Local ratings are a series of probabilistic ratings $S_{ij} = \{s_{ij}^1, s_{ij}^2, \ldots, s_{ij}^h\}, 0 \leq$

$s_{ij}^k \leq 1, h$ is bounded by the allowed history H. Therefore the local rating can be obtained by using simple averaging or exponential averaging. Equation 4.31 shows the simple averaging and Equation 4.32 shows exponential averaging. Aggregate ratings combine the local ratings with those of any witnesses (recommenders). This rating determines trustworthiness. The equations for aggregate ratings are shown in Equations 4.25 and 4.34.

Equation 4.31 adds up all the ratings from 0 to h and divides by the total number, h to get an average value.

$$R(P_i, P_j) = \begin{cases} \sum_{k=1}^{h} s_{ij}/h & h \neq 0 \\ 0 & h = 0 \end{cases} \tag{4.31}$$

For Equation 4.32, γ determines the weights given to the most recent observations. γ ranges from $(0 < \gamma < 1)$ and the larger the value the faster past observations are forgotten. The difference between the two occurs when there are malicious peers and simple averaging will give a value under the true value.

$$R(P_i, P_j) = \begin{cases} \gamma[s_{ij}^h + \ldots + (1-\gamma)^h s_{ij}^1] & h \neq 0 \\ 0 & h = 0 \end{cases} \tag{4.32}$$

P_i uses Equation 4.25 to get an aggregate rating value about P_j. $\{W_1, \ldots, W_L\}$ are a group of witnesses for peer P_j and $R(W_k, P_j)$ is the local rating. w_k is a weight assigned to testify against the credibility of the recommender (witness). $\mathcal{P}$, Equation 4.33, is a prediction from all the testimonies used to calculate the aggregate rating towards peer P_j, Equation 4.34.

$$\mathcal{P} = \begin{cases} \sum_{k=1}^{L} w_k * R(W_k, P_j)/L & L \neq 0 \\ 0.5 & L = 0 \end{cases} \tag{4.33}$$

η is peer P_i's confidence in its local rating about peer P_j where $\eta = h/H$ and L is the number of witnesses P_i found. When P_j is new, the trust rating is 0.5 because it was found that it is more advantageous to trust new peers in a p2p system as there aren't many malicious peers.

$$T(P_i, P_j) = \begin{cases} \eta R(P_i, P_j) + (1-\eta)\mathcal{P} & L \neq 0 \\ 0.5 & L = 0 \end{cases} \tag{4.34}$$

4.3.2 Ratings Discovery. Ratings discovery uses a trust graph to get referrals from other peers. The trust graph $(P_r, P_g, \mathbf{P}, \mathbf{R})$ is a directed graph composed of referral chains of P_r requesting information on P_g. $\mathbf{P}$ is a set of peers $\{P_1, \ldots, P_n\}$ and $\mathbf{R}$ is a set of referrals $\{r_1, \ldots, r_n\}$. An example trust graph is shown in Figure 4.1.

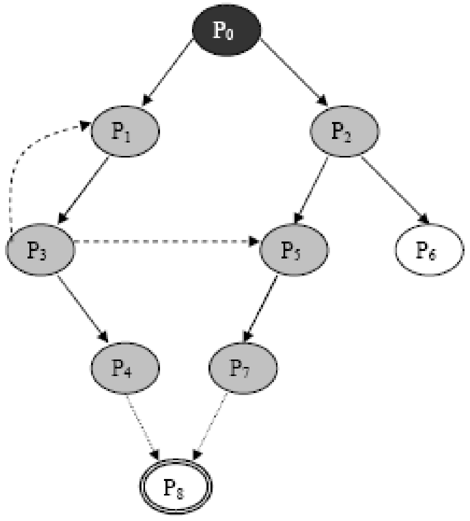

Figure 4.1: An example of a trust graph from [22].

P_0 is trying to evaluate the trustworthiness of P_8. P_4 and P_7 are witnesses (or recommenders) for P_8. The requesting peer is black, queried peers are gray, and those that have not been queried are white.

4.3.3 Ratings Aggregation. Ratings aggregation deals with incorrect ratings, noisy ratings, that distort the true ratings of peers. Equation 4.35 defines the three types of noisy ratings: complementary, exaggerated positive and exaggerated negative. α ranges from $(0 < \alpha < 1)$ and represents the exaggerated coefficient, s is the true rating and $s^{'}$ is the distorted rating. Equation 4.36 allows for the weighting of different witnesses, depending on how much a peer trusts another.

$$s^{'} = \begin{cases} 1 - s & \textit{complementary} \\ \alpha + s - \alpha s & \textit{exaggerated positive} \\ s - \alpha s/(1-\alpha) & \textit{exaggerated negative} \end{cases} \tag{4.35}$$

$$\theta = 1 - (1-\beta) \mid R(W_k, P_j) - s \mid \tag{4.36}$$

V. Experimental Methodology

The last chapter walked through gathering some functional and non-functional requirements for Cybercraft as well as exploring two use cases used to examine potential real-life scenarios Cybercraft might be exposed to. Requirements gathering is now applied to the trust models to give a self-defense guarantee and commander's trust evaluation/expression for Cybercraft. To do this, we use three specific trust models: hTrust, VTrust and P2P. The trust models give a mathematical approach to gauge the trustworthiness of interacting entities and allow for precisely evaluating security assumptions, attacks, and risks within the Cybercraft architecture. A mathematical approach gives understanding for transitive trust and root of trust questions specific to Cybercraft missions.

The focus of this chapter is to set up two scenarios for Cybercraft architecture exploration. The first scenario deals with transitive and root of trust questions and the second scenario uses the first use case of updating anti-virus software from Chapter III to analyze and investigate trust relationships for a potential real-life scenario for Cybercraft. The goal of these scenarios is to define the trust relationships within the Cybercraft domain (reference Figure 2.1). Defining these relationships will help establish what type of model is needed to give value to the trust relationships expressed in Cybercraft. For each scenario, there is an initial set of dummy values to create and populate the trust management framework for the trust models. An area for future research is defining the separate components that make up the Cybercraft domain and placing appropriate values to each. In the Cybercraft domain model, the lines represent a trust value. This trust value is created with trust models. The two scenarios are set up with three trust models: hTrust, VTrust, and P2P.

Certain characteristics must be true in order for us to use a trust model. A model should be able to form, maintain, and evolve trust opinions between entities. Quality of service (QoS) requirements are essential as they decide whether interaction or transactions will take place between interacting entities. There might not be a globally available infrastructure to interact with and a model should account for this as well. Interacting entities should be dynamic and anonymous. Finally, a model should incorporate human trust decision, be subjective and highly customizable.

The rest of the chapter is laid out as follows. Scenario one is discussed first in section 5.1, followed by scenario two in section 5.2.

5.1 Scenario One: Transitive Trust Between Entities

The goal of this scenario is to find out how far transitive trust can go. An example is $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$ which is read a trusts b who trusts c who trusts d who trusts e and therefore a trusts e. We want to see how far this trust chain can go before it falls apart. Next we walk through each trust model and how to set up all the values and formulas for this scenario.

There are two types of transitive trust: agent to agent (which Stevens did [18] and the root of trust. The root of trust is the one we are most interested in. We want to know if the root of trust in the Cybercraft, or was it the OS, transfers to other parts of the system.

5.1.1 hTrust. For hTrust, we have to set up each agents local environment. Table 5.1 represents the local environment for all agents a - e.

5.1.2 VTrust. For VTrust, we only use the recommendation component for scenario one. The values of experience and knowledge do not count as the weight to create the normalized vectors cancels them out. We break down this scenario into several cases. Case one will use the initial data setup shown in Table 5.3. Recommendation values are set to 0.9 and steady throughout a chain of 26 agents.

Table 5.1: Local Environment in scenario one - hTrust.

	b's data	c's data	d's data
Aggregated Trust Information	$[b, a, 0.3, 0.3, 4]$ $[b, c, 0.3, 0.3, 4]$	$[c, b, 0.3, 0.3, 4]$ $[c, d, 0.3, 0.3, 4]$	$[d, c, 0.3, 0.3, 4]$ $[d, e, 0.3, 0.3, 4]$
Tacit Information	$[b, a, 0.4, 0.3, 4]$ $[b, c, 0.4, 0.3, 4]$	$[c, b, 0.4, 0.3, 4]$ $[c, d, 0.4, 0.3, 4]$	$[d, c, 0.4, 0.3, 4]$ $[d, e, 0.4, 0.3, 4]$
Portfolio of Credentals	$[a, b, 0.3, 0.3, 4]_{SK_a}$ $[c, b, 0.3, 0.3, 4]_{SK_c}$	$[b, c, 0.3, 0.3, 4]_{SK_b}$ $[d, c, 0.3, 0.3, 4]_{SK_d}$	$[c, d, 0.3, 0.3, 4]_{SK_c}$ $[e, d, 0.3, 0.3, 4]_{SK_e}$
	a's data	e's data	
Aggregated Trust Information	$[a, b, 0.3, 0.3, 4]$	$[e, d, 0.3, 0.3, 4]$	
Tacit Information	$[a, b, 0.4, 0.3, 4]$	$[e, d, 0.4, 0.3, 4]$	
Portfolio of Credentials	$[b, a, 0.3, 0.3, 4]_{SK_b}$	$[d, e, 0.3, 0.3, 4]_{SK_d}$	

Table 5.2: Initial parameters for all entities in scenario one - hTrust.

Parameters	
T	2 minutes
δ_{max}	0.8
k_{min}	0.1
η	0
Customizing Functions	
h_1	(l_1', l_2')
h_2	$w_1 = 0, w_2 = 1$
h_3	$0,\ if\ l_1 < 0\ and\ l_2 > 0$ 1 otherwise
h_4	$w_1 = 1, w_2 = 1, w_3 = 0$
h_5	$l_i = \frac{n*l_i + \frac{2-\delta l_i}{2}}{n+1}\ if\ \delta l_i > l_i$ $l_i = \frac{n*l_i - \frac{2-\delta l_i}{2}}{n+1}\ if\ \delta l_i < l_i$

Table 5.4 are the weights given to each component of the vector $(A \rightarrow B)_t^N$, where $W_e + W_k + W_r = 1$. For this experiment, we only consider recommendations as we are trying to gauge how well VTrust deals with transitive trust and how far down the chain we can go.

Table 5.3: Beginning parameters in scenario one, case one - VTrust.

Trustor Initial Recommendation Values	
$(A \rightarrow B)_t^N$	0.9
$(B \rightarrow C)_t^N$	0.9
$(C \rightarrow D)_t^N$	0.9
$\vdots$	$\vdots$
$(X \rightarrow Y)_t^N$	0.9
$(Y \rightarrow Z)_t^N$	0.9

Table 5.4: Weighting of vector values, scenario one - VTrust.

Weight Values	
W_e	0
W_k	0
W_r	1

Stevens thesis [18] proved that going through a chain of 5 agents with a recommendation value of 0.9 for each link $(A \rightarrow B)$ results in a final trust value of 0.59049. In the experiments we will test different values to better evaluate what a typical scenario for a Cybercraft will be. Table 5.5 shows the walk through the trust chain and resulting values.

Table 5.5: Results from chain of 5, scenario one - VTrust.

Recommendation Chain	
$A \rightarrow C$	0.81
$A \rightarrow D$	0.73
$A \rightarrow E$	0.6561
$A \rightarrow F$	0.59049

Case two puts a little bit more realistic values to start. Table 5.6 shows the initial values. Each agent represents a part of the Cybercraft domain. Agent A represents a Cybercraft platform, agent B is a payload, agent C represents an OS, agent D the network, agent E a different OS, agent F a payload on this separate machine and agent G is the locally installed Cybercraft platform. If a payload goes through a formal verification, it follows that the platform and payload will have complete trust and knowledge between each other.

Table 5.6: Beginning parameters in scenario one, case two - VTrust.

Trustor Initial Recommendation Values	
$(A \rightarrow B)_t^N$	1.0
$(B \rightarrow C)_t^N$	0.8
$(C \rightarrow D)_t^N$	0.2
$(D \rightarrow E)_t^N$	0.2
$(E \rightarrow F)_t^N$	0.8
$(F \rightarrow G)_t^N$	1.0

5.1.3 P2P. The P2P model uses certain assumptions on the system. First, there are numerous peers entering and leaving all the time and second, malicious peers are rare in this type of system. For the first scenario, trust chaining, certain parameters need to be defined and set. Table 5.7 shows the parameters and values given for this scenario. The bound of referral chain's length (D, the ** value in the table) is what is being evaluated in this scenario and thus is not set. The setup is similar to the two previous models. Peer P_1 trusts peer P_2 at 0.9, peer P_2 trusts P_3 at 0.9, and so on. For case two, P2P uses the same values as shown in Table 5.6.

5.2 Scenario Two: Anti-virus Update

Scenario two uses the fully dressed use case from Chapter III, section 3.2, that deals with AV software. There are three cases: main successful scenario, AV is not

Table 5.7: Parameters in scenario one - P2P.

Symbol	Value	Description
h	1	Number of latest interactions
H	10	Bound of allowed history
D	**	Bound of referral chain's length
B	2	Branching factor
α	0.1	Exaggeration coefficient
β	0.5	Constant
γ	0.5	Averaging constant
θ	-	Update factor in Equation 4.36
η	h/H	Confidence about local ratings
σ_i	0.5	Threshold of referral generation
ω_i	0.5	Threshold of trust

Table 5.8: Beginning values in scenario one - P2P.

Symbol	Value	Description
$R(W_1, P_6)$	0.9	Local rating of witness W_1 for peer P_6
w_i	1	Weight for the credibility of witness W_1
η	0.1	-

installed, and AV is not updated. Each model uses a set of dummy variables to start with. An assumption we use is payloads cannot talk to each other. They must go through the Cybercraft platform and the platform will talk to the other payload. The general agents are defined in Table 5.9.

Table 5.9: General agent setup for all trust models, scenario two.

Agent	**Value**
A	Cybercraft platform
B	Cybercraft payload check
C	Cybercraft payload update
D	Cybercraft payload install
E	OS
F	Network
G	AV software on OS (agent E)
H	Update place
I	AV software from network

Case one: Cybercraft platform creates a payload, ‘payload check’, to check if there is AV software loaded on the machine and, if so, is up-to-date. The payload must interact with the OS to inquire about AV software. The OS queries its applications and responds to the payload that yes, there is indeed AV software loaded. The payload then queries the AV software to ensure it is up-to-date. AV software responds that everything is good to go. Finally, the payload reports back to the Cybercraft platform that AV software is installed and up-to-date on this particular OS.

Case two: Cybercraft platform creates a payload, ‘payload check’, to check if there is AV software loaded on the machine, and, if so, is up-to-date. The payload interacts with the OS to inquire about the AV software. The OS queries its applications and find there is no AV software loaded and reports these findings back to the payload. The payload reports to the Cybercraft platform there is no AV software loaded on this OS. The Cybercraft platform creates a new payload, ‘payload install’, to go find the AV software. The new payload must interact with the OS and network to get to the AV software that is on the network. Once the payload gets the AV software, it interacts with the OS to install the AV software. Once installed, the payload (install) reports back to the Cybercraft platform that AV software is installed. The Cybercraft platform then dispatches the previous payload (check) to ensure the newly installed AV software is up-to-date. The payload interacts with the OS again to query the AV software. The AV software responds to the payload everything is up-to-date. The payload (check) reports to the Cybercraft platform everything is up-to-date for the AV software on this particular machine OS.

Case three: Cybercraft platform creates a payload, ‘payload check’, to check if there is AV software loaded on the machine, and, if so, is up-to-date. The payload interacts with the OS to inquire about the AV software. The OS queries its applications and responds to the payload there is AV software loaded. The payload queries the AV software to ensure it is up-to-date. The AV software responds that no, it is not up-to-date. The payload then reports to the Cybercraft platform that the AV software loaded on the OS is not up-to-date. The Cybercraft platform creates a new payload, ‘payload update’, to update the AV software. The new payload must interact with the OS and network to get to the AV software update on the network. Once the payload gets the AV update , it interacts with the OS to install the update. Once the update is installed, the payload (update) reports back to the Cybercraft platform that AV software is up-to-date.

Certain assumptions were made for this scenario. The first is that all parts of the system are secure and trustworthy (not malicious). Network, OS, Cybercraft platform and agent. Next, all trust interactions result in a good trust value. Lastly, the goal of the scenario is to evaluate the resulting values of trust between each of the three models.

5.2.1 hTrust. For hTrust, there is an initial set of tuples for the agents that know about each other. Walking through the cases, we figure out where all those equations fit in. Each agent wanting to talk to another agent it does not know about must run the recommendation protocol (reference Chapter II, Section 2.6.1.2), which uses the trust formation function Υ (Equation 4.12) to create a trust prediction. For the sake of these cases, we always choose the middle value. After the interaction, tuples are exchanged between the two agents and each runs the aggregation function Φ (reference Equation 4.16) to update their tuples in their local environment, if any. Then, the requesting agent, the trustor, will run the tacit information extraction function Ψ (reference Equation 4.21) to update the recommendation value of the agent that recommended the trustee.

The parameters are similar as in scenario one, except a few minor changes, shown in Table 5.10. The weights for the customizing function, h_2 are both 0.5 as we want to consider trust reflexivity and trust transitivity. Everything else is left the same as from scenario one. The next sections go through the data setup for each case in scenario two for hTrust.

The trust values were chosen for each agent as close to what a real scenario would look like. For example, the Cybercraft platform, agent a, will trust a payload explicitly if it has been formally verified and that platform created it. The Cybercraft payload check (agent b) has a trust level of 1 and knowledge 1, but the Cybercraft payloads update and install (c and d) have a trust level of 0.5 (knowledge still 1 because the Cybercraft platform created them) because these payloads will be interacting with other parts of the system and therefore will have a lower initial trust value. The OS (agent e) will have full trust and knowledge of items currently installed on it, such as the AV software (agent g). The OS does not directly correspond with the Cybercraft platform (agent a), and therefore does not have full trust in it, as well as the network.

Table 5.10: Initial parameters for all entities in scenario two.

Parameters	
T	2 minutes
δ_{max}	0.8
k_{min}	0.1
η	0
Customizing Functions	
h_1	$l_1^{'}, l_2^{'})$
h_2	$w_1 = 0.5, w_2 = 0.5$
h_3	$0,\ if\ l_1 < 0\ and\ l_2 > 0$
	1 otherwise
h_4	$w_1 = 1, w_2 = 1, w_3 = 0$
h_5	$l_i = \frac{n*l_i + \frac{2-\delta l_i}{2}}{n+1}\ if\ \delta l_i > l_i$
	$l_i = \frac{n*l_i - \frac{2-\delta l_i}{2}}{n+1}\ if\ \delta l_i < l_i$

5.2.1.1 *Case One.*

Case one uses agents a, b, e and g. Time starts at $t = 10$ at the beginning of the scenario to account for the generation of trust through time. Table 5.11 shows the local environment of all the agents.

5.2.1.2 *Case Two.*

Case two uses agents a, b, d, e, f and i. Time starts at $t = 10$ at the beginning of the scenario to account for the generation of trust through time. Table 5.12 displays the agents' local environment.

Table 5.11: Local environment for scenario two, case one - hTrust.

	a's data	e's data
Aggregated Trust Information	$[a, b, 1, 1, 10]$ $[a, e, 0.7, 0.5, 10]$	$[e, a, 0.8, 0.8, 10]$ $[e, g, 1, 1, 10]$
Tacit Information	$[a, b, 0.8, 1, 10]$ $[a, e, 0.5, 0.5, 10]$	$[e, a, 0.8, 0.8, 10]$ $[e, g, 1, 1, 10]$
Portfolio of Credentials	$[b, a, 1, 1, 10]_{SK_b}$ $[e, a, 0.8, 0.8, 10]_{SK_e}$	$[a, e, 0.7, 0.5, 10]_{SK_a}$ $[g, e, 1, 1, 10]_{SK_g}$
	b's data	g's data
Aggregated Trust Information	$[b, a, 1, 1, 10]$	$[g, e, 1, 1, 10]$
Tacit Information	$[b, a, 1, 1, 10]$	$[g, e, 1, 1, 10]$
Portfolio of Credentials	$[a, b, 1, 1, 10]_{SK_a}$	$[e, g, 1, 1, 10]_{SK_e}$

Table 5.12: Local environment for scenario two, case two - hTrust.

	a's data	e's data	f's data
Aggregated Trust Information	$[a, b, 1, 1, 10]$ $[a, d, 0.5, 1, 10]$ $[a, e, 0.75, 0.5, 10]$	$[e, a, 0.8, 0.8, 10]$ $[e, f, 0.8, 0.8, 10]$	$[f, e, 0.8, 0.8, 10]$ $[f, i, 0.8, 0.8, 10]$
Tacit Information	$[a, b, 0.8, 1, 10]$ $[a, d, 0.5, 1, 10]$ $[a, e, 0.5, 0.5, 10]$	$[e, a, 0.8, 0.8, 10]$ $[e, f, 0.8, 0.8, 10]$	$[f, e, 0.8, 0.8, 10]$ $[f, i, 0.8, 0.8, 10]$
Portfolio of Credentials	$[b, a, 1, 1, 10]_{SK_b}$ $[d, a, 1, 1, 10]_{SK_d}$ $[e, a, 0.8, 0.8, 10]_{SK_e}$	$[a, e, 0.7, 0.5, 10]_{SK_a}$ $[f, e, 0.8, 0.8, 10]_{SK_f}$	$[e, f, 0.8, 0.8, 10]_{SK_e}$ $[i, f, 0.8, 0.8, 10]_{SK_i}$
	b's data	d's data	i's data
Aggregated Trust Information	$[b, a, 1, 1, 10]$	$[d, a, 1, 1, 10]$	$[i, f, 0.8, 0.8, 10]$
Tacit Information	$[b, a, 1, 1, 10]$	$[d, a, 1, 1, 10]$	$[i, f, 0.8, 0.8, 10]$
Portfolio of Credentials	$[a, b, 1, 1, 10]_{SK_a}$	$[a, d, 0.5, 1, 10]_{SK_a}$	$[f, i, 0.8, 0.8, 10]_{SK_f}$

5.2.1.3 Case Three. Case three uses agents a, b, c, e, f, g and h. Time starts at $t = 10$ at the beginning of the scenario to account for the generation of trust through time. Table 5.13 is the local environment for the agents.

Table 5.13: Local environment for scenario two, case three - hTrust.

	a's data	e's data	f's data
Aggregated Trust Information	$[a, b, 1, 1, 10]$ $[a, c, 1, 1, 10]$ $[a, e, 0.7, 0.5, 10]$	$[e, a, 0.8, 0.8, 10]$ $[e, f, 0.8, 0.8, 10]$ $[e, g, 1, 1, 10]$	$[f, e, 0.8, 0.8, 10]$ $[f, h, 0.8, 0.8, 10]$
Tacit Information	$[a, b, 0.8, 1, 10]$ $[a, c, 0.9, 1, 10]$ $[a, e, 0.5, 0.5, 10]$	$[e, a, 0.8, 0.8, 10]$ $[e, f, 0.8, 0.8, 10]$ $[e, g, 1, 1, 10]$	$[f, e, 0.8, 0.8, 10]$ $[f, h, 0.8, 0.8, 10]$
Portfolio of Credentials	$[b, a, 1, 1, 10]_{SK_b}$ $[c, a, 1, 1, 10]_{SK_c}$ $[e, a, 0.8, 0.8, 10]_{SK_e}$	$[a, e, 0.7, 0.5, 10]_{SK_a}$ $[f, e, 0.8, 0.8, 10]_{SK_f}$ $[g, e, 1, 1, 10]_{SK_g}$	$[e, f, 0.8, 0.8, 10]_{SK_e}$ $[h, f, 0.8, 0.8, 10]_{SK_h}$
	b's data	c's data	g's data
Aggregated Trust Information	$[b, a, 1, 1, 10]$	$[c, a, 1, 1, 10]$	$[g, e, 1, 1, 10]$
Tacit Information	$[b, a, 1, 1, 10]$	$[c, a, 1, 1, 10]$	$[g, e, 1, 1, 10]$
Portfolio of Credentials	$[a, b, 1, 1, 10]_{SK_a}$	$[a, c, 1, 1, 10]_{SK_a}$	$[e, g, 1, 1, 10]_{SK_e}$
	h's data		
Aggregated Trust Information	$[h, f, 0.8, 0.8, 10]$		
Tacit Information	$[h, f, 0.8, 0.8, 10]$		
Portfolio of Credentials	$[f, h, 0.8, 0.8, 10]_{SK_f}$		

5.2.2 VTrust. VTrust differs from hTrust in that the trust vectors are not updated after each interaction, but from a series of events. The experience component is computed by taking a series of events, broken into time intervals, and giving a weight value to each set. The time interval set for this case is $ti = 2$. $[t_0, t_1]$ represents one interval. There was no initial setting to the vectors for this scenario. Parameters for case two are shown in table 5.14.

Table 5.14: Parameters for VTrust, scenario two.

Parameters	
dk_w	Direct knowledge weight
ik_w	Indirect knowledge weight
E_w	Experience component weight
K_w	Knowledge component weight
R_w	Recommendation component weight
d	Direct knowledge value
r	Indirect knowledge value

5.2.2.1 Case One. Case one uses agents A, B, E and G. Time starts at $t = 0$ at the beginning of the scenario to account for the generation of trust through time. Table 5.15 is the local environment for the agents.

Table 5.15: Local environment in scenario two, case one - VTrust.

A's environment		B's environment					
$A - B$		$B - A$		$B - E$		$B - G$	
dk_w	1	dk_w	1	dk_w	0.7	dk_w	0.7
ik_w	0	ik_w	0	ik_w	0.3	ik_w	0.3
E_w	0.5	E_w	0.5	E_w	0.5	E_w	0.5
K_w	0.5	K_w	0.5	K_w	0.3	K_w	0.3
R_w	0	R_w	0	R_w	0.2	R_w	0.2
d	1	d	0.3	d	0.3	d	1
r	0	r	0	r	0.7	r	0.6
E's environment				G's environment			
$E - B$		$E - G$		$G - B$		$G - E$	
dk_w	0.7	dk_w	1	dk_w	0.7	dk_w	1
ik_w	0.3	ik_w	0	ik_w	0.3	ik_w	0
E_w	0.5	E_w	0.5	E_w	0.5	E_w	0.5
K_w	0.3	K_w	0.5	K_w	0.3	K_w	0.5
R_w	0.2	R_w	0	R_w	0.2	R_w	0
d	0.3	d	1	d	0.3	d	1
r	0.7	r	0	r	0.7	r	0

5.2.2.2 Case Two. Case two uses agents A, B, D, E, F and I. Time starts at $t = 0$ at the beginning of the scenario to account for the generation of trust through time. Table 5.16 is the agents' local environment.

5.2.2.3 Case Three. Case three uses agents A, B, C, E, F, G and H. Time starts at $t = 0$ at the beginning of the scenario to account for the generation of trust through time. Table 5.17 is the local environment of the agents.

Table 5.17: Local environment in scenario two, case three - VTrust.

C's environment									
$C - A$		$C - E$		$C - F$		$C - H$		$C - G$	
dk_w	1	dk_w	0.7	dk_w	0.7	dk_w	0.7	dk_w	0.7
ik_w	0	ik_w	0.3	ik_w	0.3	ik_w	0.3	ik_w	0.3
E_w	0.5	E_w	0.5	E_w	0.5	E_w	0.5	E_w	0.5
K_w	0.5	K_w	0.3	K_w	0.3	K_w	0.3	K_w	0.3
R_w	0	R_w	0.2	R_w	0.2	R_w	0.2	R_w	0.2
d	1	d	0.3	d	0.3	d	0.3	d	0.3
r	0	r	0.7	r	0.5	r	0.6	r	0.6
A's environment				B's environment					
$A - B$		$A - C$		$B - E$		$B - A$		$B - G$	
dk_w	1	dk_w	1	dk_w	0.7	dk_w	1	dk_w	0.7
ik_w	0	ik_w	0	ik_w	0.3	ik_w	0	ik_w	0.3
E_w	0.5	E_w	0.5	E_w	0.5	E_w	0.5	E_w	0.5
K_w	0.5	K_w	0.5	K_w	0.3	K_w	0.5	K_w	0.3
R_w	0	R_w	0	R_w	0.2	R_w	0	R_w	0.2

continued on next page

Table 5.16: Local environment in scenario two, case two - VTrust.

A's environment				B's environment			
$A-B$		$A-D$		$B-E$		$B-A$	
dk_w	1	dk_w	1	dk_w	0.7	dk_w	1
ik_w	0	ik_w	0	ik_w	0.3	ik_w	0
E_w	0.5	E_w	0.5	E_w	0.5	E_w	0.5
K_w	0.5	K_w	0.5	K_w	0.3	K_w	0.5
R_w	0	R_w	0	R_w	0.2	R_w	0
d	1	d	1	d	0.3	d	1
r	0	r	0	r	0.7	r	0
D's environment							
$D-A$		$D-E$		$D-F$		$D-I$	
dk_w	1	dk_w	0.7	dk_w	0.7	dk_w	0.7
ik_w	0	ik_w	0.3	ik_w	0.3	ik_w	0.3
E_w	0.5	E_w	0.5	E_w	0.5	E_w	0.5
K_w	0.5	K_w	0.3	K_w	0.3	K_w	0.3
R_w	0	R_w	0.2	R_w	0.2	R_w	0.2
d	1	d	0.3	d	0.3	d	0.3
r	0	r	0.7	r	0.5	r	0.6
E's environment				F's environment			
$E-B$		$E-D$		$F-D$		$F-I$	
dk_w	0.7	dk_w	0.7	dk_w	0.7	dk_w	1
ik_w	0.3	ik_w	0.3	ik_w	0.3	ik_w	0
E_w	0.5	E_w	0.5	E_w	0.5	E_w	0.5
K_w	0.3	K_w	0.3	K_w	0.3	K_w	0.5
R_w	0.2	R_w	0.2	R_w	0.2	R_w	0
d	0.3	d	0.3	d	0.3	d	1
r	0.7	r	0.7	r	0.8	r	0
I's environment							
$I-D$		$I-F$					
dk_w	0.7	dk_w	1				
ik_w	0.3	ik_w	0				
E_w	0.5	E_w	0.5				
K_w	0.3	K_w	0.5				
R_w	0.2	R_w	0				
d	0.3	d	1				
r	0.8	r	0				

continued from previous page									
d	1	d	1	d	0.3	d	1	d	0.3
continued on next page									

continued from previous page									
r	0	r	0	r	0.7	r	0	r	0.6
H's environment				F's environment					
$H-C$		$H-F$		$F-E$		$F-H$		$F-C$	
dk_w	0.7	dk_w	1	dk_w	01	dk_w	1	dk_w	0.7
ik_w	0.3	ik_w	0	ik_w	0	ik_w	0	ik_w	0.3
E_w	0.5	E_w	0.5	E_w	0.5	E_w	0.5	E_w	0.5
K_w	0.3	K_w	0.5	K_w	0.5	K_w	0.5	K_w	0.3
R_w	0.2	R_w	0	R_w	0	R_w	0	R_w	0.2
d	0.3	d	1	d	1	d	1	d	0.3
r	0.8	r	0	r	0	r	0	r	0.8
E's environment									
$E-B$		$E-G$		$E-C$		$E-F$			
dk_w	0.7	dk_w	1	dk_w	0.7	dk_w	1		
ik_w	0.3	ik_w	0	ik_w	0.3	ik_w	0		
E_w	0.5	E_w	0.5	E_w	0.5	E_w	0.5		
K_w	0.3	K_w	0.5	K_w	0.3	K_w	0.5		
R_w	0.2	R_w	0	R_w	0.2	R_w	0		
d	0.3	d	1	d	0.3	d	1		
r	0.7	r	1	r	0.7	r	1		
G's environment									
$G-E$		$G-B$		$G-C$					
dk_w	01	dk_w	0.7	dk_w	0.7				
ik_w	0	ik_w	0.3	ik_w	0.3				
E_w	0.5	E_w	0.5	E_w	0.5				
K_w	0.5	K_w	0.3	K_w	0.3				
continued on next page									

continued from previous page

R_w	0	R_w	0.2	R_w	0.2				
d	1	d	0.3	d	0.3				
r	0	r	0.7	r	0.7				

5.2.3 P2P. The parameters are very similar to those of scenario one. Table 5.18 shows the parameters and values given for this scenario.

Table 5.18: Parameters for the P2P model, scenario one.

Symbol	Value	Description
β	0.5	Constant
θ	-	Update factor in Equation 4.36
η	.5	Confidence about local ratings

5.2.3.1 Case One. Case one uses agents P_a, P_b, P_e and P_g. Time starts at $t = 0$ at the beginning of the scenario to account for the generation of trust through time. The only initial values of this scenario are the fact that P_a and P_b, and P_e and P_g have a direct knowledge of 1 for each other.

5.2.3.2 Case Two. Case two uses agents P_a, P_b, P_d, P_e, P_f, and P_i. Time starts at $t = 0$ at the beginning of the scenario to account for the generation of trust through time. The only initial values of this scenario are the fact that P_a and P_b, P_a and P_d, P_e and P_f, and P_e and P_i have a direct knowledge of 1 for each other.

5.2.3.3 Case Three. Case three uses peers $P_a, P_b, P_c, P_e, P_f, P_g$ and P_h. Time starts at $t = 0$ at the beginning of the scenario to account for the generation of trust through time. The only initial values of this scenario are the fact that P_a and P_b, P_a and P_c, P_e and P_f, P_f and P_h, and P_e and P_g have a direct knowledge of 1 for each other.

VI. Analysis and Results

This chapter covers the results of the scenarios from each of the three models. Scenario one examined two variations of transitive trust: agent to agent and the root of trust. Scenaro two used the AV use case from chapter III, section 3.2. Scenario one results are discussed in section 6.1 and scenario two results are discussed in section 6.2.

6.1 Scenario One: Transitive Trust

6.1.1 Analysis. The two types of transitive trust analyzed are agent to agent and the root of trust. Stevens [18] work focused mainly on transitive trust between agents.

6.1.2 Results. We breakout the results of scenario one based on our three candidate trust models. We discuss hTrust first, then VTrust, and conclude with P2P analysis.

6.1.2.1 hTrust. The results show that the chain trust from $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$ falls apart after c. hTrust has no transitivity of trust. Agent a receives a recommendation from b about c and a's knowledge of c is set to 0. To continue with the chain, there must be some sort of interaction between a and c for a to have knowledge about c as a recommender. A way for things to work would be for the knowledge component to be transitive just as the trust value is. Case two was not evaluated because of the results from case one.

6.1.2.2 VTrust. For case one, setting all values to 0.9 for recommendations, agent's $A - Z$ create a chain that, with the range from [0,1], still does not go to 0. This is an unrealistic scenario as there is a very small chance there will be a set of agents that have a chain with a recommendation value of 0.9 for every chain. The results are shown in Figure 6.1 and show that even through 26 agents, the trust value is not negative and not quite 0 (0.0646). There is a 0.75 difference in the beginning value and last trust recommendation.

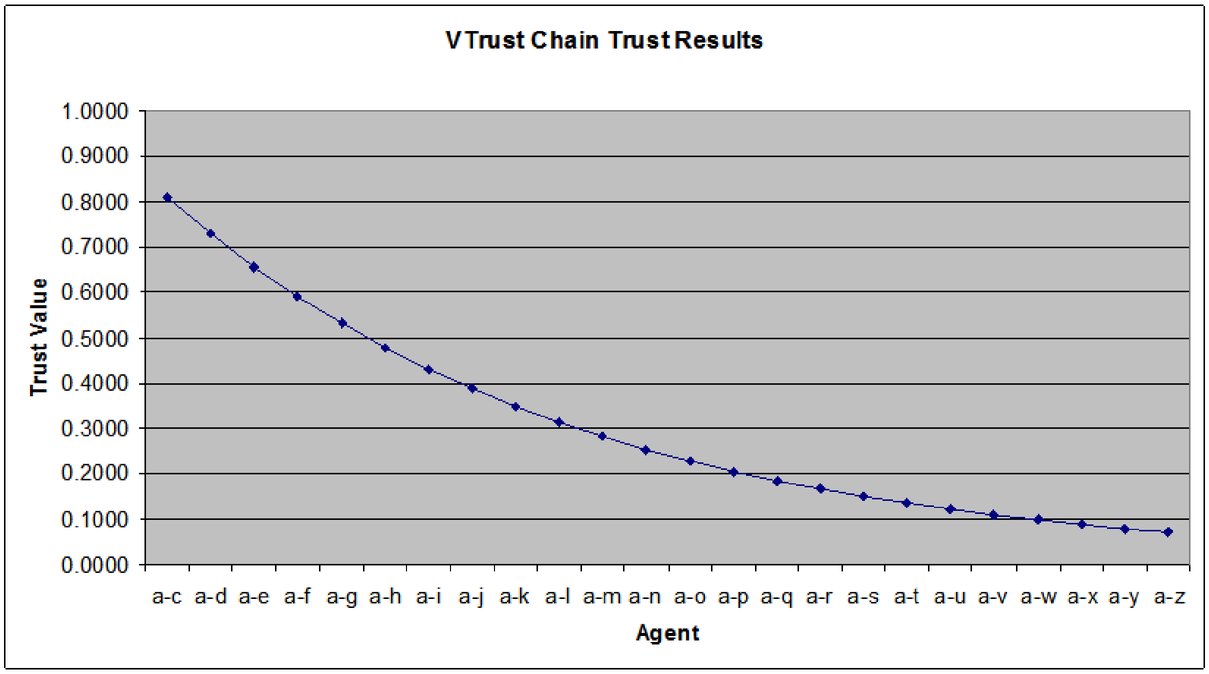

Figure 6.1: Results for trust chaining agents $A - Z$ - VTrust.

The second case uses more realistic values mapping to a possible Cybercraft domain and trys to more accurately predict the environment a Cybercraft will be in. The results, from Figure 6.2, show trust degrades much faster. It only takes two recommendations before the value drops 0.6 points, which is very significant, especially within the range $\in$ [0,1].

6.1.2.3 P2P. The P2P model shows that there is a high limit (unknown as of yet) for trust chaining. Because of the way a P2P system is setup, peers trust until proven otherwise. Therefore the trust chain resulting from peer P_i attempting to interact with peer P_j, can be very long. P_i uses the witness(es) that attest to P_j. The chain is used to find these witnesses and nothing more. The witnesses' (W_k) local rating for P_j is used, along with a weighting of the trust P_i has in W_k (set to 1 if P_i has never interacted with W_k before) to get a predicted trust value. If this value is above a certain threshold, ω, then peer P_i will interact with P_j and update

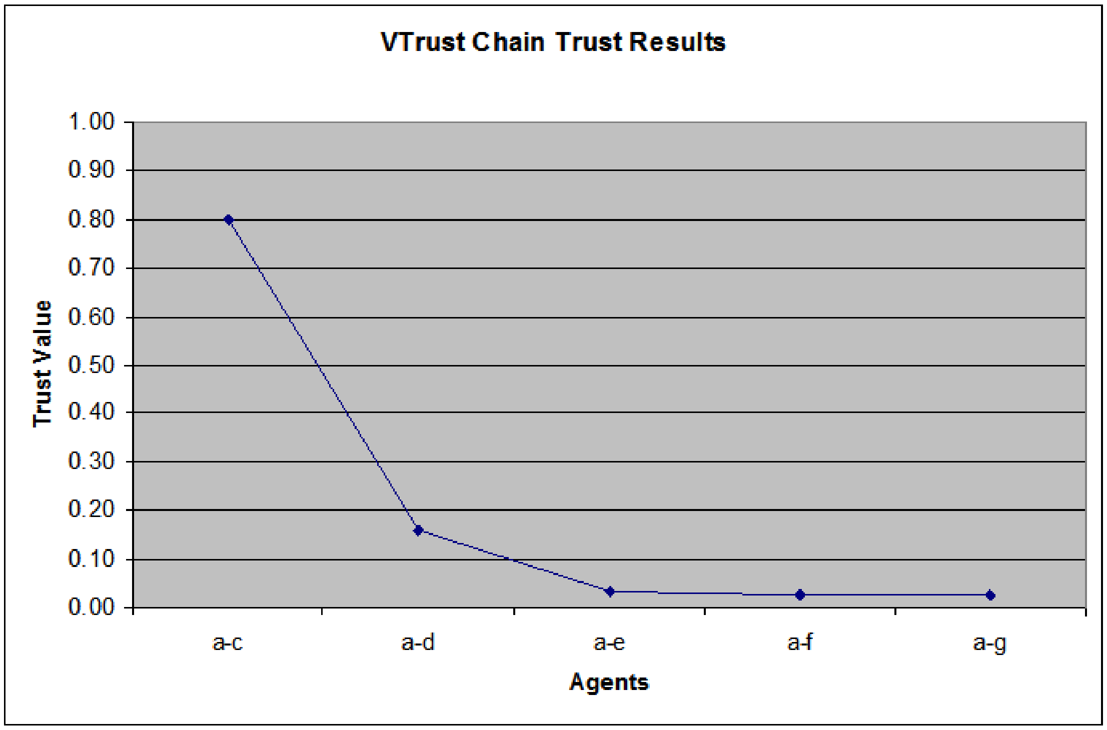

Figure 6.2: Results for trust chaining agents $A - G$ - VTrust.

its information. For our first scenario, we used a value of 0.9. This yielded a final trust value of 0.81, with a threshold of 0.5. Therefore, peer P_i will interact with peer P_j. A summary of the results is shown in Table 6.1. Changing the beginning value will not affect the resulting trust value in a significant way

Table 6.1: Results from the P2P model, scenario one, case one.

Recommendation Chain Results	
$P_1 \rightarrow P_6$	0.81

Case two uses the same values as VTrust in scenario one, case two. Peer P_1 wishes to interact with peer P_7 and the resulting trust value is 0.05. The reson for this is because of the value $P_6 \rightarrow P_7 = 0.1$. Thus, peer P_1 uses that value as a starting point. The results are shown in Table 6.2.

Table 6.2: Results from the P2P model, scenario one, case two.

Recommendation Chain Results	
$P_1 \rightarrow P_6$	0.05

6.2 Scenario Two: Anti-virus Update

6.2.1 Analysis. The goal for scenario two was to implement one of the fully-dressed use cases from Chapter III, specifically the one dealing with AV update.

6.2.2 Results. The results for case one, two and three are shown in Figures 6.3, 6.4 and 6.5. The graph shows the agent interactions on the x-axis. It is read agent a trusts agent b ($a - b$). The y-axis is the resulting trust value at the end of the scenario. The results are similar through all three cases. The numbers are very similar between each model and agent. The difference comes in where hTrust has a 1 value throughout the entire scenario. hTrust therefore does not allow for the factor of time decay. The other two models factored that in and thus their numbers are lower for those agent trust relationships. Another observation is hTrust generally has a lower trust value for certain agents (ones that didn't start with a 1 value) than the other two models VTrust and P2P.

For hTrust, time started at t = 10, increased in increments of 2, and ended at t = 16. The interactions for case one, in order, were $a-b$, $b-e$, $e-g$, $g-e$, $e-b$, $b-a$. Agents $b-g$ and $e-g$ did not exchange any sort of recommendations and thus their tacit information tuples were not updated. The results from show that with somewhat realistic values, trust will not degrade too fast to a point where the model becomes useless. A case for future work is to run scenarios adjusting all the parameters (such as the customizing functions $h_1 - h_4, \eta$, etc.). For case two, time started at t = 10, increased in increments of 2, and ended at t = 22. The interactions, in order, were $a-b, b-e$, $e-b$, $b-a$, $a-d$, $d-e$, $d-f$, $d-i$, $i-d$, $d-e$, $d-a$, $a-b$, $b-i$, $i-b$, $b-a$. Agents $e-f$ and $f-i$ did not exchange any sort of recommendations and thus their tacit information tuples were not updated. For case three time started at t = 10, increased in increments of 2, and ended at t = 20. The interactions, in order, were $a-b$, $b-e$, $e-g$, $e-b$, $b-a$, $a-c$, $c-e$, $c-f$, $c-h$, $h-c$, $c-e$, $c-a$, $a-c$. Agents $e-g$ and $f-h$ did not exchange any sort of recommendations and thus their tacit information tuples were not updated.

VTrust calculates the value of trust over a period of time as opposed to after each interaction (such as hTrust). For this scenario, all interactions were recorded, then a final trust value was created. The P2P model calcutes trust values after each interaction.

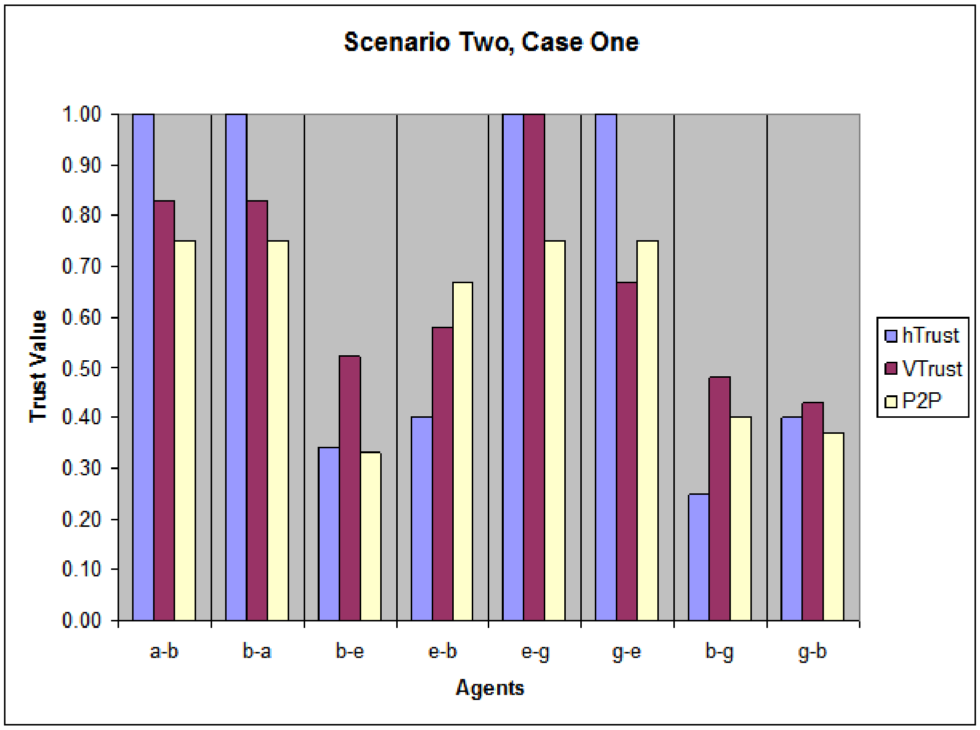

Figure 6.3: Results for Scenario Two, Case One.

6.3 Reference Framework

A proponent of hTrust is the various parameters and customizing functions. These elements allow for a better evaluation of trust. The problem with hTrust is the model breaks down after only one recommendation. hTrust does not in its current state allow for the degredation of trust over time. Cybercraft are going to need to have

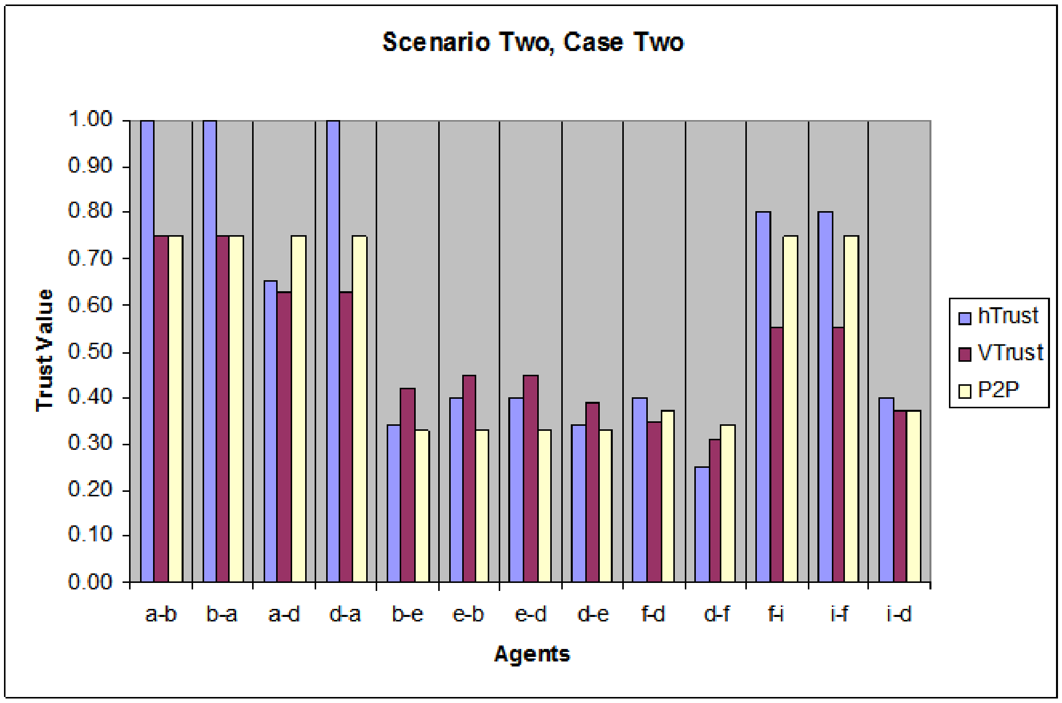

Figure 6.4: Results for Scenario Two, Case Two.

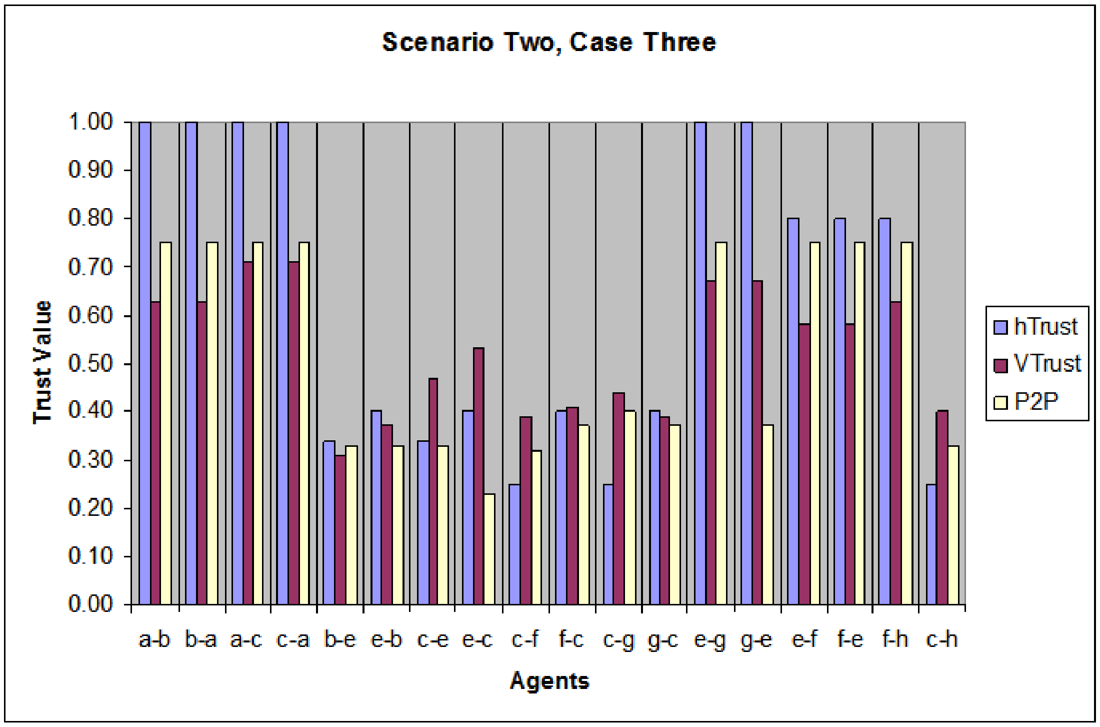

Figure 6.5: Results for Scenario Two, Case Three.

numerous chains. VTrust allows for trust chaining but does not have a framework to implement the model and seems to be a piecemeal of components to create a value. P2P allows for a long chain of trust but with the assumption to trust a peer until it is proven malicious will not work for Cybercraft. The opposite is true, Cybercraft must be less trusting of new peers. Another thought is P2P only allows for an interval of [0,1]. This is not adequate to represent a full range of trust values. Recommend a range of [-1,1] such as that of hTrust.

None of these models taken as-is will work for Cybercraft. With some modifications, a combination of the best of all three, or even a new model, is a possibility.

Table 6.3: Reference Framework.

	hTrust	VTrust	P2P
Able to form, maintain, and evolve trust opinions	Yes	Yes	Yes
Incorporates QoS	Yes	Yes	Yes
Human tailored	Yes	No	No
Subjective	Yes	Yes	Yes
Highly customizable	Yes	No	Yes
Allows for transitive trust	No	Yes	Yes
Dynamic trust changing	Yes	Yes	Yes
Minimal resource demands	Yes	Yes	Yes
Ability to catch malicious agents	Yes	Yes	Yes
Allow for degredation of trust over time	No	Yes	Yes

VII. Conclusions and Future Work

Trust expression in the Cybercraft domain centers around several key relationships: platform to payload, platform to platform, platform to node, and payload to payload. There are obviously many others to consider. Our continuing work considers which, if any, of the current trust models best express proof that trust transfers from the root of trust in the payload to other components in the system.

There are many unknowns still to be captured and analyzed for Cybercraft. The goal for this research effort is not a network defense band aid for USAF networks, but rather a means to focus thinking about future threats and capabilities. In this paper, we discuss various means of capturing requirements, specifically using the software engineering techniques of use cases, threat modeling, and attack trees. As well, the main research question of trust was discussed. We expound briefly possible trust models such as hTrust and VTrust that may provide a good fit for Cybercraft. Future work in this are includes iterative requirements exposition for Cybercraft and enumeration of trust relationships within the Cybercraft domain.

7.1 Future Work

Cybercraft is a new idea for the future of our network defense and has many unknowns still. The following section discusses various work that will help further the goal of implementing Cybercraft to protect our defense networks.

7.1.1 Requirements. TCNO's and TCTO's are an excellent source for gathering future requirements. These documents contain the daily guidelines for our current network and should be used in helping define what our future network will be like. Another area in requirements gathering is using the brief use cases from chapter III and creating fully-dressed use cases. Once these use cases are created, trust model application is the next step. Addressing and evaluating the current state of our network defense is important to ensure we address any deficiencies there might be.

7.1.2 Trust. The many components that make up the Cybercraft domain will have values to represent trust. Future work can decide what value to give a trusted vendor versus unknown vendors or vendors that did not go through a screening process (formal verification). As well as vendors, USAF networks extensively use third party software, such as A/V software, firewall, etc. There needs to be a way to place a value of trust in these components. Communication lines must be secure and trusted as well and given a trust value. One idea for placing initial trust values is if the product offered goes through a formal verification process, then the trust value is 1, if not, it could be 0 or possibly lower, even to -1. More trust models need to be evaluated to further define the reference framework for the Cybercraft domain. Evaluating the models using the same scenarios and adjusting the parameters will help pin down what values are realistic for those variables.

Appendix A. Operational-Tactical/Mission Breakdown

The following tables are the result of the requirements analysis from Chapter III. The analysis of these tables are included in Chapter III.

Table A.2 shows the brief use cases created for the defense priority of automated attack interdiction from Chapter III.

Table A.1: Attack Detection operational-tactical/mission breakdown.

Strategic goal: Attack Detection		
Operational	**Tactical**	**Mission**
Detect Insider Threat	Monitor access	Ensure access is as limited as possible
		Ensure a "need to know"
		Log individual access to highly sensitive items
		Ensure clearances are up-to-date
		Examine physical security to devices (server farm, unattended computers, wiring closets, etc.)
		Maintain password policies
		Deactivate access immediately after termination
		Monitor and respond to suspicious behavior
		Record user access 24X7
Detect physical threat	Monitor access	Log individual access to highly sensitive items
		Examine physical security to devices (server farm, unattended computers, wiring closets, etc.)
Detect computer attack	Determine source of attack	Monitor incoming IP addresses
		Monitor incoming/outgoing ports
		Monitor high activity sources
Detection analysis	Detect anomalies	Analyze base traffic for XX months
		Compare incidents over XX amount of time

Table A.2: Automated Network Vulnerability Mitigation operational-tactical/mission breakdown.

Strategic goal: Automated Network Vulnerability Mitigation		
Operational	**Tactical**	**Mission**
Detect and mitigate Denial of Service (DDoS) attacks	Monitor collective network resource utilization and deny attempts to usurp XX% utilization	Monitor CPU historical usage
		Monitor server storage growth
		Monitor router/switch activity
		Monitor/analyze base incoming network activity
		Ensure global parameters are set correctly
		Collect and analyze data on network performance
		Monitor overall network status, check for anomalies
		Monitor specific base network status (i.e., a specific base, unit, or squadron)
	Daily/weekly scheduled maintenance	Log system crashes
		Ensure regular backups are occurring
		Use tools to detect configuration changes

continued on next page

continued from previous page

Strategic goal: Automated Network Vulnerability Mitigation		
Operational	**Tactical**	**Mission**
		Ensure all patches are up to date on routers, switches, workstations, etc.
		System defragmentation
		Scheduled outage/downtime
	Check vulnerabilities	Ensure global parameters are set correctly
		Ensure proper configuration of server, workstation, peripherals, communications devices, and OS/application software
		Ensure network redundancy
		Ensure "hot swap" devices are available
		Enable router filtering of known DDoS attacks
Ensure workstations, network devices are up-to-date	Automated patching	Ensure SMS is enabled
		Ensure minimal exemption list

Table A.3: Automated Attack Interdiction operational-tactical/mission breakdown.

Strategic goal: Automated Attack Interdiction		
Operational	**Tactical**	**Mission**
Detect network attack	Detect and mitigate base external network resources	Monitor log files
		Block specific IP segments attack is coming from
		Ensure router/switch ACL's are up-to-date
		Cut off from network compromised devices
	detect and mitigate base internal network resources	Monitor log files
		Ensure all patches are up-to-date
Automatic blocking	Block ports	Start blocking a "suspect" port when traffic reaches XX% utilization
		Block unused ports
		Allow only certain traffic on ports well known for attacks
Active deception	Redirection	Redirect to honeypots
Documentation	Logs	Monitor logs
		Detect anomalies

Table A.4: Network Attack Damage Assessment operational-tactical/mission breakdown.

Strategic goal: Network Attack Damage Assessment		
Operational	**Tactical**	**Mission**
Determine network baseline	Monitor network	Log network traffic for XX months to create baseline
		Create baseline ACL's for routers and switches
		Implement standard desktop

Table A.5: Automated Attack Reporting operational-tactical/mission breakdown.

Strategic goal: Automated Attack Reporting		
Operational	**Tactical**	**Mission**
Documentation		Track incidents

Table A.6: Adversary Identification operational-tactical/mission breakdown.

Strategic goal: Adversary Identification		
Operational	**Tactical**	**Mission**
		Track IP addresses/segments of known attacks
		Analyze logs for trends

Appendix B. MFR 2008-01-17

This is a MFR from Mr. Lou Giannelli, INOSE East Det 3. We conducted weekly meetings to hash out some possible Cybercraft missions using attack/defense trees. This MFR is a result of a few examples Mr. Giannelli brought up during one of our meetings.

Subject: Cybercraft Suggestions

Submitted by: Lou Giannelli, INOSC East Det 3

Date: 17 Jan, 2008

B.1 Background

As a sub product of the weekly session interaction where Defenders have provided Capt Hunt with defense and attack tree scenarios, it has become apparent that the Defenders can also provide suggestions for Cybercraft capabilities. This MFR documents two suggestions presented to Capt Hunt on today's session. These suggestions represent possible solutions to mitigate two recurrent problems.

B.2 Problem 1

Defenders routinely provide base technicians with technical details regarding suspicious/malicious network connections. The said details are provided as raw data transcripts in ASCII format. The average technician customarily is not prepared to identify key elements necessary to identify and validate questionable connections.

Suggested solution 1. To study the feasibility of enabling Cybercraft with a parsing subroutine capable of enabling base technicians with a search utility to locate these key elements in a raw data transcript in ASCII format.

B.3 Problem 2

There are recurrent violations to current AFI and TCNO policy. The violations originate on base internal users circumventing sercurity policies.

Suggested solution 2. To study the feasibility of enabling Cybercraft with a logical subroutine capable of flagging an alert on outbound connections using the listed services when a set of conditions are met.

This suggestion envisions Cybercraft enabled with a logical subroutine capable of examining the 3-way handshake sequence on packets with destination port 21, and a destination IP different than the authorized sites. Cybercraft will flag connections meeting these criteria with an alert.

Appendix C. Results

This appendix includes all the results from the scenarios in Chapter VI.

Table C.1: Results from a chain of 26 agents.

Recommendation Chain Results					
$A \to C$	0.81	$A \to M$	0.2824	$A \to W$	0.0985
$A \to D$	0.729	$A \to N$	0.2542	$A \to X$	0.0886
$A \to E$	0.6561	$A \to O$	0.2288	$A \to Y$	0.0798
$A \to F$	0.59059	$A \to P$	0.2059	$A \to Z$	0.0718
$A \to G$	0.5314	$A \to Q$	0.1853		
$A \to H$	0.4783	$A \to R$	0.1668		
$A \to I$	0.4305	$A \to S$	0.1501		
$A \to J$	0.3874	$A \to T$	0.1351		
$A \to K$	0.3487	$A \to U$	0.1216		
$A \to L$	0.3138	$A \to V$	0.1094		

Table C.2: Results from a more realistic Cybercraft environment.

Recommendation Chain Results	
$A \to C$	0.80
$A \to D$	0.16
$A \to E$	0.032
$A \to F$	0.0256
$A \to G$	0.0256

Table C.3: Final Data for scenario two, case one - hTrust.

a's data	
Aggregated Trust Information	$[a, b, 1, 1, 16]$
Tacit Information	$[a, b, 1, 1, 10]$
Portfolio of Credentials	$[b, a, 1, 1, 16]_{SK_b}$
b's data	
Aggregated Trust Information	$[b, a, 1, 1, 16]$ $[b, e, 0.34, 0.1, 12]$ $[b, g, 0.25, 0.1, 14]$
Tacit Information	$[b, a, 1, 0.97, 12]$ $[b, e, 0.83, 0.1, 14]$
Portfolio of Credentials	$[a, b, 1, 1, 16]_{SK_a}$ $[e, b, 0.4, 0.1, 12]_{SK_e}$ $[g, b, 0.4, 0.1, 14]_{SK_g}$
e's data	
Aggregated Trust Information	$[e, g, 1, 1, 12]$ $[e, b, 0.4, 0.1, 12]$
Tacit Information	$[e, g, 0.97, 1, 12]$
Portfolio of Credentials	$[g, e, 1, 1, 12]_{SK_g}$ $[b, e, 0.34, 0.1, 12]_{SK_b}$
g's data	
Aggregated Trust Information	$[g, e, 0.97, 1, 12]$ $[g, b, 0.4, 0.1, 14]$
Tacit Information	$[g, e, 1, 1, 12]$
Portfolio of Credentials	$[e, g, 1, 1, 10]_{SK_e}$ $[b, g, 0.25, 0.1, 14]_{SK_b}$

Table C.4: Final Data for scenario two, case two - hTrust.

	a's data	b's data	d's data
Aggregated Trust Information	$[a, b, 1, 1, 22]$ $[a, d, 0.65, 1, 20]$	$[b, a, 1, 1, 22]$ $[b, e, 0.34, 0.1, 12]$ $[b, i, 0.13, 0.1, 22]$	$[d, a, 1, 1, 20]$ $[d, e, 0.34, 0.2, 18]$ $[d, i, 0.25, 0.1, 18]$ $[d, f, 0.25, 0.1, 16]$
Tacit Information	$[a, b, 1, 1, 10]$ $[a, d, 0.5, 1, 10]$	$[b, a, 0.97, 1, 12]$	$[d, a, 0.99, 1, 14]$ $[d, e, 0.75, 0.1, 16]$ $[d, e, 0.75, 0.1, 18]$
Portfolio of Credentials	$[b, a, 1, 1, 22]_{SK_b}$ $[d, a, 1, 1, 20]_{SK_d}$	$[a, b, 1, 1, 22]_{SK_a}$ $[e, b, 0.4, 0.1, 12]_{SK_e}$ $[i, b, 0.4, 0.1, 22]_{SK_i}$	$[a, d, 0.65, 1, 20]_{SK_a}$ $[e, d, 0.4, 0.2, 18]_{SK_e}$ $[i, d, 0.4, 0.1, 18]_{SK_i}$ $[f, d, 0.4, 0.1, 16]_{SK_f}$
	e's data	f's data	i's data
Aggregated Trust Information	$[e, b, 0.4, 0.1, 12]$ $[e, d, 0.4, 0.2, 18]$ $[e, i, 0.25, 0.1, 18]$	$[f, d, 0.4, 0.1, 16]$	$[i, b, 0.4, 0.1, 12]$ $[i, d, 0.4, 0.1, 18]$ $[i, e, 0.4, 0.1, 18]$
Tacit Information	$[e, g, 0.97, 1, 12]$	$[f, e, 1, 1, 12]$	
Portfolio of Credentials	$[b, e, 0.34, 0.1, 12]_{SK_b}$ $[d, e, 0.25, 0.1, 18]_{SK_d}$ $[i, e, 0.25, 0.1, 18]_{SK_i}$	$[d, f, 0.25, 0.1, 16]_{SK_e}$	$[b, i, 0.13, 0.1, 22]_{SK_b}$ $[d, i, 0.25, 0.1, 18]_{SK_d}$ $[e, i, 0.25, 0.1, 18]_{SK_e}$

Table C.5: Final Data for scenario two, case three - hTrust.

	a's data	b's data	c's data
Aggregated Trust Information	$[a, b, 1, 1, 22]$ $[a, d, 0.65, 1, 20]$	$[b, a, 1, 1, 12]$ $[b, e, 0.34, 0.1, 12]$	$[c, a, 1, 1, 20]$ $[c, e, 0.34, 0.1, 14]$ $[c, f, 0.25, 0.1, 16]$ $[c, g, 0.25, 0.1, 20]$ $[c, h, 0.25, 0.1, 18]$
Tacit Information	$[a, b, 1, 1, 12]$ $[a, c, 1, 1, 20]$	$[b, a, 0.97, 1, 12]$	$[c, a, 0.99, 1, 14]$ $[c, e, 0.75, 0.1, 16]$ $[c, f, 0.75, 0.1, 18]$
Portfolio of Credentials	$[b, a, 1, 1, 12]_{SK_b}$ $[c, a, 1, 1, 20]_{SK_c}$	$[a, b, 1, 1, 12]_{SK_a}$ $[e, b, 0.4, 0.1, 12]_{SK_e}$	$[a, c, 1, 1, 20]_{SK_a}$ $[e, c, 0.4, 0.1, 14]_{SK_e}$ $[f, c, 0.4, 0.1, 16]_{SK_f}$ $[g, c, 0.4, 0.1, 20]_{SK_g}$ $[h, c, 0.4, 0.1, 18]_{SK_g}$
	e's data	f's data	g's data
Aggregated Trust Information	$[e, b, 0.4, 0.1, 12]$ $[e, c, 0.4, 0.1, 14]$ $[e, g, 1, 1, 12]$	$[f, c, 0.4, 0.1, 16]$	$[g, c, 0.4, 0.1, 20]$ $[g, e, 1, 1, 12]$
Tacit Information	$[e, g, 0.97, 1, 12]$	$[f, e, 1, 1, 12]$	
Portfolio of Credentials	$[b, e, 0.34, 0.1, 12]_{SK_b}$ $[c, e, 0.34, 0.1, 14]_{SK_c}$ $[g, e, 1, 1, 12]_{SK_g}$	$[c, f, 0.25, 0.1, 16]_{SK_c}$	$[c, g, 0.25, 0.1, 20]_{SK_c}$ $[e, c, 0.4, 0.1, 14]_{SK_e}$
	h's data		
Aggregated Trust Information	$[h, c, 0.4, 0.1, 18]$		
Tacit Information	(none)		
Portfolio of Credentials	$[c, h, 0.25, 0.1, 18]_{SK_c}$		

Table C.6: Final data for scenario two, case one - VTrust.

A's data	
$(A \rightarrow B)$	[0.33, 0.5, 0.0]
Trust value	0.83
B's data	
$(B \rightarrow A)$	[0.33, 0.5, 0.0]
Trust value	0.83
$(B \rightarrow E)$	[0.25, 0.13, 0.14]
Trust value	0.52
$(B \rightarrow G)$	[0.17, 0.12, 0.2]
Trust value	0.48
E's data	
$(E \rightarrow B)$	[0.25, 0.13, 0.2]
Trust value	0.58
$(E \rightarrow G)$	[0.5, 0.5, 0.0]
Trust value	1
G's data	
$(G \rightarrow B)$	[0.17, 0.13, 0.14]
Trust value	0.43
$(G \rightarrow E)$	[0.17, 0.5, 0.0]
Trust value	0.67

Table C.7: Final data for scenario two, case two - VTrust.

A's data		*B*'s data	
$(A \rightarrow B)$	[0.25, 0.5, 0.0]	$(B \rightarrow A)$	[0.25, 0.5, 0.0]
Trust value	0.75	Trust value	0.75
$(A \rightarrow D)$	[0.13, 0.5, 0.0]	$(B \rightarrow E)$	[0.15, 0.13, 0.14]
Trust value	0.63	Trust value	0.42
F's data		*I*'s data	
$(F \rightarrow D)$	[0.05, 0.14, 0.16]	$(I \rightarrow D)$	[0.08, 0.14, 0.16]
Trust value	0.35	Trust value	0.37
$(F \rightarrow I)$	[0.05, 0.5, 0.0]	$(I \rightarrow F)$	[0.05, 0.5, 0.0]
Trust value	0.55	Trust value	0.55
D's data		*E*'s data	
$(D \rightarrow A)$	[0.13, 0.5, 0.0]	$(E \rightarrow B)$	[0.13, 0.13, 0.2]
Trust value	0.63	Trust value	0.45
$(D \rightarrow E)$	[0.13, 0.13, 0.14]	$(E \rightarrow D)$	[0.13, 0.13, 0.2]
Trust value	0.39	Trust value	0.45
$(D \rightarrow F)$	[0.5, 0.11, 0.15]		
Trust value	0.31		
$(D \rightarrow I)$	[0.08, 0.12, 0.16]		
Trust value	0.35		

Table C.8: Final data for scenario two, case three - VTrust.

A's data		C's data	
$(A \rightarrow B)$	[0.13, 0.5, 0.0]	$(C \rightarrow A)$	[0.21, 0.5, 0.0]
Trust value	0.63	Trust value	0.71
$(A \rightarrow C)$	[0.21, 0.5, 0.0]	$(C \rightarrow E)$	[0.21, 0.13, 0.14]
Trust value	0.71	Trust value	0.47
B's data			
$(B \rightarrow A)$	[0.13, 0.5, 0.0]	$(C \rightarrow F)$	[0.13, 0.11, 0.16]
Trust value	0.63	Trust value	0.39
$(B \rightarrow E)$	[0.04, 0.13, 0.14]	$(C \rightarrow H)$	[0.13, 0.12, 0.16]
Trust value	0.31	Trust value	0.4
$(B \rightarrow G)$	[0.04, 0.12, 0.2]	$(C \rightarrow G)$	[0.13, 0.12, 0.0]
Trust value	0.36	Trust value	0.44
F's data		G's data	
$(F \rightarrow E)$	[0.08, 0.5, 0.0]	$(G \rightarrow E)$	[0.17, 0.5, 0.0]
Trust value	0.58	Trust value	0.67
$(F \rightarrow H)$	[0.13, 0.5, 0.0]	$(G \rightarrow B)$	[0.04, 0.13, 0.14]
Trust value	0.63	Trust value	0.31
$(F \rightarrow C)$	[0.13, 0.14, 0.15]	$(G \rightarrow C)$	[0.13, 0.13, 0.14]
Trust value	0.41	Trust value	0.39
E's data		H's data	
$(E \rightarrow B)$	[0.04, 0.13, 0.2]	$(H \rightarrow F)$	[0.13, 0.5, 0.0]
Trust value	0.37	Trust value	0.63
$(E \rightarrow G)$	[0.17, 0.5, 0.2]	$(H \rightarrow C)$	[0.13, 0.14, 0.16]
Trust value	0.67	Trust value	0.42
$(E \rightarrow C)$	[0.21, 0.13, 0.2]		
Trust value	0.53		
$(E \rightarrow F)$	[0.08, 0.5, 0.0]		
Trust value	0.58		

Table C.9: Final data for scenario two, case one - P2P.

P_e's data			P_g's data		
$P_e - P_b$	$R(P_e, P_b)$	0.0	$P_g - P_e$	$R(P_g, P_e)$	1.0
	$T(P_e, P_b)$	0.67		$T(P_g, P_e)$	0.75
$P_e - P_g$	$R(P_e, P_g)$	1.0	$P_g - P_b$	$R(P_g, P_b)$	0.0
	$T(P_e, P_g)$	0.75		$T(P_g, P_b)$	0.37
P_b's data			P_a's data		
$P_b - P_a$	$R(P_b, P_a)$	1.0	$P_a - P_b$	$R(P_a, P_b)$	1.0
	$T(P_b, P_a)$	0.75		$T(P_a, P_b)$	0.75
$P_b - P_e$	$R(P_b, P_e)$	0.0			
	$T(P_b, P_e)$	0.33			
$P_b - P_g$	$R(P_b, P_g)$	0.0			
	$T(P_b, P_g)$	0.4			

Table C.10: Final data for scenario two, case two - P2P.

P_a's data			P_b's data		
$P_a - P_b$	$R(P_a, P_b)$	1.0	$P_b - P_a$	$R(P_b, P_a)$	1.0
	$T(P_a, P_b)$	0.75		$T(P_b, P_a)$	0.75
$P_a - P_d$	$R(P_a, P_d)$	1.0	$P_b - P_e$	$R(P_b, P_e)$	0.0
	$T(P_a, P_d)$	0.75		$T(P_b, P_e)$	0.33
P_e's data			P_f's data		
$P_e - P_b$	$R(P_e, P_b)$	0.0	$P_f - P_d$	$R(P_f, P_d)$	0.0
	$T(P_e, P_b)$	0.33		$T(P_f, P_d)$	0.37
$P_e - P_d$	$R(P_e, P_d)$	0.0	$P_f - P_i$	$R(P_f, P_i)$	1.0
	$T(P_e, P_d)$	0.33		$T(P_f, P_i)$	0.75
P_d's data			P_i's data		
$P_d - P_a$	$R(P_d, P_a)$	1.0	$P_i - P_d$	$R(P_i, P_d)$	0.0
	$T(P_d, P_a)$	0.75		$T(P_i, P_d)$	0.37
$P_d - P_e$	$R(P_d, P_e)$	0.0	$P_i - P_f$	$R(P_i, P_f)$	1.0
	$T(P_d, P_e)$	0.33		$T(P_i, P_f)$	0.75
$P_d - P_f$	$R(P_d, P_f)$	0.0			
	$T(P_d, P_f)$	0.34			
$P_d - P_i$	$R(P_d, P_i)$	0.0			
	$T(P_d, P_i)$	0.36			

Table C.11: Final data for scenario two, case three - P2P.

P_a's data			P_h's data		
$P_a - P_b$	$R(P_a, P_b)$	1.0	$P_h - P_f$	$R(P_h, P_f)$	1.0
	$T(P_a, P_b)$	0.75		$T(P_h, P_f)$	0.75
$P_a - P_c$	$R(P_a, P_c)$	1.0	$P_h - P_c$	$R(P_h, P_c)$	0.0
	$T(P_a, P_c)$	0.75		$T(P_h, P_c)$	0.37
P_b's data			P_f's data		
$P_b - P_a$	$R(P_b, P_a)$	1.0	$P_f - P_e$	$R(P_f, P_e)$	1.0
	$T(P_b, P_a)$	0.75		$T(P_f, P_e)$	0.75
$P_b - P_e$	$R(P_b, P_e)$	0.0	$P_f - P_h$	$R(P_f, P_h)$	1.0
	$T(P_b, P_e)$	0.33		$T(P_f, P_h)$	0.75
$P_b - P_g$	$R(P_b, P_g)$	0.0	$P_f - P_c$	$R(P_f, P_c)$	0.0
	$T(P_b, P_g)$	0.33		$T(P_f, P_c)$	0.37
P_c's data			P_e's data		
$P_c - P_a$	$R(P_c, P_a)$	1.0	$P_e - P_b$	$R(P_e, P_b)$	0.0
	$T(P_c, P_a)$	0.75		$T(P_e, P_b)$	0.33
$P_c - P_e$	$R(P_c, P_e)$	0.0	$P_e - P_g$	$R(P_e, P_g)$	1.0
	$T(P_c, P_e)$	0.33		$T(P_e, P_g)$	0.75
$P_c - P_f$	$R(P_c, P_f)$	0.0	$P_e - P_c$	$R(P_e, P_c)$	0.0
	$T(P_c, P_f)$	0.32		$T(P_e, P_c)$	0.23
$P_c - P_h$	$R(P_c, P_h)$	0.0	$P_e - P_f$	$R(P_e, P_f)$	1.0
	$T(P_c, P_h)$	0.33		$T(P_e, P_f)$	0.75
$P_c - P_g$	$R(P_c, P_g)$	0.0			
	$T(P_c, P_g)$	0.4			
P_g's data					
$P_g - P_e$	$R(P_g, P_e)$	1.0			
	$T(P_g, P_e)$	0.75			
$P_g - P_b$	$R(P_g, P_b)$	0.0			
	$T(P_g, P_b)$	0.33			
$P_g - P_c$	$R(P_g, P_c)$	0.0			
	$T(P_g, P_c)$	0.37			

Bibliography

1. AFRL/RYT. "AFRL/RYT (Anti-Tamper and Software Protection)", 2008. URL `http://www.wpafb.af.mil/library/factsheets/factsheet.asp?id=6308`.
2. Air Force Research Laboratory, Information Directorate. Conference slides from Cyber Defense Conference, 2007.
3. Bishop, Matt. *Computer Security: Art and Science*. Addison Wesley, 2003.
4. Cahill, Vinny, Brian Shand, Colin English, Giovanna di Marzo Serugendo, and Marco Carbone. "Using Trust for Secure Collaboration in Uncertain Environments". *IEEE Pervasive Computing*, 2(3):52, 2003.
5. Capra, Licia. "Engineering human trust in mobile system collaborations". *SIGSOFT '04/FSE-12: Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering*, 107–116. ACM Press, New York, NY, USA, 2004. ISBN 1-58113-855-5.
6. Edge, Kenneth S. *A Framework for Analyzing and Mitigating the Vulnerabilities of Complex Systems via Attack and Protection Trees*. Ph.D. thesis, Air Force Institute of Technology, 2007.
7. Esfandiari, B. and S. Chandrasekharan. "On How Agents Make Friends: Mechanisms for Trust Acquisition", 2001. URL `citeseer.ist.psu.edu/esfandiari01how.html`.
8. Gambetta, Diego. *Can We Trust Trust?* Basil Blackwell, 1988. URL `citeseer.ist.psu.edu/gambetta88can.html`. Reprinted in electronic edition from Department of Sociology, University of Oxford, chapter 13, pp. 213-237".
9. Gettle, Mitch. "Air Force releases new mision statement", December 2005. URL `http://www.af.mil/news/story.asp?id=123013440`.
10. Gollmann, Dieter. "Why Trust is Bad for Security". *First International Workshop on Security and Trust Management*, 3–9. European Research Consortium in Informatics and Mathematics, September 2005.
11. Karrels, Daniel R. and Gilbert Peterson. "CyberCraft: Protecting Air Force Electronic Systems with Lightweight Agents". Vir V. Phoha and S.S. Iyengar (editors), *Proceedings of the Cyberspace Research Workshop*, 58–62. United States Air Force, Shreveport, LA, November 2007.
12. Larman, Craig. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Prentice Hall PTR, 3 edition, 2005.
13. Lopez, C. Todd. "8th Air Force to become new cyber command", November 2006. URL `http://www.af.mil/news/story.asp?storyID=123030505`.

14. McDonald, Jeffrey and Shannon Hunt. "Developing a Requirements Framework for Cybercraft Trust Evaluation". *Proceedings of the 3rd International Conference on Information Warfare and Security*. 2008.

15. Phister, Dr Paul Jr., Dan Fayette, and Emily Krzysiak. "CyberCraft: Concept Linking NCW Principles with the Cyber Domain in an Urban Operational Environment". 2006.

16. Ray, Indrajit and Sudip Chakraborty. "A Vector Model of Trust for Developing Trustworthy Systems". *Proceedings of 9th European Symposium on Research in Computer Security (ESORICS'04*. Sophia Antipolis, France, 2004.

17. Ray, Indrajit, Sudip Chakraborty, and Indrakshi Ray. "VTrust: A Trust Management System Based on a Vector Model of Trust". *Proceedings of 1st International Conference on Information Systems Security (ICISS'05)*. Sophia Antipolis, France, 2005.

18. Stevens, Michael. *Use of Trust Vectors in Support of the CyberCraft Initiative*. Master's thesis, Air Force Institute of Technology, 2007.

19. Stevens, Michael and Paul D. Williams PhD. "Use of Trust Vectors for CyberCraft and the Limits of Usable Data History for Trust Vectors". *CISDA 2007 IEEE Symposium on Computational Intelligence in Security and Defense Applications*, 193–200.

20. Thompson, Ken. "Reflections on Trusting Trust". *Communications of the ACM*, 27(8):761–763, 1984.

21. Trusted Computing Group. "TCG Specification Architecture Overview", April 2007.

22. Yu, Bin, Munindar P. Singh, and Katia Sycara. "Developing Trust in Large-Scale Peer-to-Peer Systems". *Proceedings of First IEEE Symposium on Multi-Agent Security and Survivability*, 1–10. 2004.

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704–0188*

1. REPORT DATE *(DD–MM–YYYY)* 27–03–2008	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* May 2006 — Mar 2008

4. TITLE AND SUBTITLE Developing a Reference Framework for Cybercraft Trust Evaluation	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Shannon Hunt, Capt, USAF	**5d. PROJECT NUMBER** 08-198
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management, AFIT/EN 2950 Hobson Way WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GCS/ENG/08-11
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Ms. Sonja Glumich, Sonja.Glumich@rl.af.mil, 315-330-4370 Air Force Research Laboratory Information Directorate 525 Brooks Rd, Rome, New York 13441	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RIGA
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

It should be no surprise that Department of Defense (DoD) and U.S. Air Force (USAF) networks are the target of constant attack. As a result, network defense remains a high priority for cyber warriors. On the technical side, trust issues for a comprehensive end-to-end network defense solution are abundant and involve multiple layers of complexity. The Air Force Research Labs (AFRL) is currently investigating the feasibility of a holistic approach to network defense, called Cybercraft. We envision Cybercraft to be trusted computer entities that cooperate with other Cybercraft to provide autonomous and responsive network defense services. A top research goal related to Cybercraft centers around how we may examine and ultimately prove features related to this root of trust.
In this work, we investigate use-case scenarios for Cybercraft operation with a view towards analyzing and expressing trust requirements inherent in the environment. Based on a limited subset of functional requirements for Cybercraft in terms of their role, we consider how current trust models may be used to answer various questions of trust between components. We characterize generic model components that assist in answering questions regarding Cybercraft trust and pose relevant comparison criteria as evaluation points for various (existing) trust models. The contribution of this research is a framework for comparing trust models that are applicable to similar network-based architectures. Ultimately, we provide a reference evaluation framework for how (current and future) trust models may be developed or integrated into the Cybercraft architecture.

15. SUBJECT TERMS

trees, model, network architecture, requirements

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Lt Col Todd McDonald
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U	UU	97	(937) 255–3636, jeffrey.mcdonald@afit.edu